

FFFFFFFFFFFFFFFF	DDDDDDDDDDDDDD	LLL
FFFFFFFFFFFFFFFF	DDDDDDDDDDDDDD	LLL
FFFFFFFFFFFFFFFF	DDDDDDDDDDDDDD	LLL
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFFFFFFFFFFFFFF	DDD	DDD
FFFFFFFFFFFFFFF	DDD	DDD
FFFFFFFFFFFFFFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDD	DDD
FFF	DDDDDDDDDDDDDD	LLLLLLLLLLLLLLLL
FFF	DDDDDDDDDDDDDD	LLLLLLLLLLLLLLLL
FFF	DDDDDDDDDDDDDD	LLLLLLLLLLLLLLLL

```

FFFFFFFFF  DDDDDDD  LL      DDDDDDD  RRRRRRR  IIIIII  VV      VV  EEEEEEEEE  RRRRRRR
FFFFFFFFF  DDDDDDD  LL      DDDDDDD  RRRRRRR  IIIIII  VV      VV  EEEEEEEEE  RRRRRRR
FF          DD      LL      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      LL      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      LL      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      LL      DD          RR      RR      II      VV      VV  EE          RR
FFFFFFFFF  DD      DD      DD          RRRRRRR  II      VV      VV  EEEEEEE  RRRRRRR
FFFFFFFFF  DD      DD      DD          RRRRRRR  II      VV      VV  EEEEEEE  RRRRRRR
FF          DD      DD      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      DD      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      DD      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      DD      DD          RR      RR      II      VV      VV  EE          RR
FF          DD      DD      DD          RR      RR      II      VV      VV  EE          RR
DDDDDDDD  DDDDDDD  LLLLLLLLL  DDDDDDD  RR      RR      IIIIII  VV      VV  EEEEEEEEE  RR
DDDDDDDD  DDDDDDD  LLLLLLLLL  DDDDDDD  RR      RR      IIIIII  VV      VV  EEEEEEEEE  RR
                                     ....
                                     ....
                                     ....
                                     ....

LL          IIIIII  SSSSSSSS
LL          IIIIII  SSSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SSSSSS
LL          II      SSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```



```
0001 0 %TITLE 'FDLDRIVER'
0002 0 %SBTTL 'FDL Parse Table Drivers'
0003 0 MODULE FDLDRIVER ( IDENT='V04-000',
0004 0 ADDRESSING_MODE ( EXTERNAL = GENERAL ),
0005 0 ADDRESSING_MODE ( NONEXTERNAL = GENERAL ),
0006 0 OPTLEVEL=3
0007 0 ) =
0008 1 BEGIN
0009 1
0010 1 *****
0011 1 *
0012 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0013 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0014 1 * ALL RIGHTS RESERVED.
0015 1 *
0016 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0017 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0018 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0019 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0020 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0021 1 * TRANSFERRED.
0022 1 *
0023 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0024 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0025 1 * CORPORATION.
0026 1 *
0027 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0028 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0029 1 *
0030 1 *
0031 1 *****
0032 1
0033 1 ++
0034 1
0035 1 Facility: RMS-32 FDL Utilities
0036 1
0037 1 Abstract:
0038 1
0039 1 Contents:
0040 1 GET_LINE
0041 1 UPCASE
0042 1 SET_LINE
0043 1 SET_TERM
0044 1 SET_PRIMARY
0045 1 SET_SECONDARY
0046 1 START_STR
0047 1 END_STR
0048 1 SET_DATE_TIME
0049 1 SET_COMMENT
0050 1 SYNTAX_ERROR
0051 1 ERROR_CHK
0052 1 NEGATE
0053 1 SET_BLANK
0054 1 CLR_BLANK
0055 1 FDL$$READ_ERROR
0056 1 RMS_ERROR
0057 1 RMS_OPEN_ERROR
```

FDLDRIVER
V04-000

FDLDRIVER
FDL Parse Table Drivers

M 8
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 2
(1)

: 58
: 59
: 60
: 61
: 62
: 63

0058 1 |
0059 1 | Environment:
0060 1 |
0061 1 |
0062 1 |
0063 1 | --

VAX/VMS Operating System


```

65 0064 1
66 0065 1 Author: Keith B Thompson Creation date: January-1981
67 0066 1
68 0067 1
69 0068 1
70 0069 1 Modified by:
71 0070 1 V03-012 KFH0009 Ken Henderson 23 Aug 1983
72 0071 1 Fix to FDL$$GET_LINE to allow null
73 0072 1 FDL spec string.
74 0073 1 Fix calls to GET_VM and FREE_VM.
75 0074 1
76 0075 1 V03-011 KFH0008 Ken Henderson 10 Aug 1983
77 0076 1 Fixes to END_STR and SET_DATE_TIME
78 0077 1 Addition of EXTRACT_QUOTE routine
79 0078 1 Addition of TRIM_LEADING routine
80 0079 1
81 0080 1 V03-010 KFH0007 Ken Henderson 29 Jul 1983
82 0081 1 Check status of LIB$TPARSE call
83 0082 1
84 0083 1 V03-009 KFH0006 Ken Henderson 26 Apr 1983
85 0084 1 Fixed call to $BINTIM
86 0085 1
87 0086 1 V03-008 KFH0005 Ken Henderson 30 Dec 1982
88 0087 1 Fixed broken branches
89 0088 1
90 0089 1 V03-007 KFH0004 Ken Henderson 21 Dec 1982
91 0090 1 Fixed signal of FDL$_UNSECKW
92 0091 1
93 0092 1 V03-006 KFH0003 Ken Henderson 15-Nov-1982
94 0093 1 Added support for default and
95 0094 1 main parses
96 0095 1 Added support for more than 32
97 0096 1 secondaries per primary
98 0097 1 Added move to .FDL$GL_STNUMPTR of
99 0098 1 .FDL$GL_STMNTNUM
100 0099 1
101 0100 1 V03-005 KFH0002 Ken Henderson 6-Oct-1982
102 0101 1 Removed numtype
103 0102 1 Added support for ACL primary
104 0103 1 Added support for FDL STRINGS
105 0104 1
106 0105 1 V03-004 KFH0001 Ken F. Henderson 28-Jul-1982
107 0106 1 Changed terminator character from "/" to "\"
108 0107 1
109 0108 1 V03-003 KBT0067 Keith B. Thompson 23-Jun-1982
110 0109 1 Add support for multiple keywords per line
111 0110 1
112 0111 1 V03-002 KBT0029 Keith Thompson 30-Mar-1982
113 0112 1 Add upcase/lowercase processing and date/time routine
114 0113 1
115 0114 1 V03-001 KBT0019 Keith Thompson 22-Mar-1982
116 0115 1 Fix error message processing
117 0116 1
118 0117 1 ****
```

```
: 120 0118 1
: 121 0119 1 PSECT
: 122 0120 1 OWN = FDL$OWN (PIC),
: 123 0121 1 GLOBAL = FDL$GLOBAL (PIC),
: 124 0122 1 PLIT = FDL$PLIT (SHARE,PIC),
: 125 0123 1 CODE = FDL$CODE (SHARE,PIC);
: 126 0124 1
: 127 0125 1 LIBRARY 'SYSSLIBRARY:STARLET';
: 128 0126 1 REQUIRE 'SRC$:FDLUTIL';
: 129 0311 1 REQUIRE 'LIB$:FDLPARDEF';
: 130 0850 1
: 131 0851 1 EXTERNAL ROUTINE
: 132 0852 1 FDL$$GET_VM,
: 133 0853 1 FDL$$FREE_VM,
: 134 0854 1 LIB$TPARSE,
: 135 0855 1 STR$TRIM,
: 136 0856 1 SYSS$BINTIM;
: 137 0857 1
: 138 0858 1 DEFINE_ERROR_CODES;
: 139 0859 1
: 140 0860 1 FORWARD ROUTINE
: 141 0861 1 EXTRACT_QUOTE,
: 142 0862 1 TRIM_LEADING,
: 143 0863 1 UPCASE : NOVALUE,
: 144 0864 1 FDL$$SET_PRIMARY,
: 145 0865 1 FDL$$SET_SECONDARY,
: 146 0866 1 FDL$$START_STR,
: 147 0867 1 FDL$$END_STR,
: 148 0868 1 FDL$$SET_COMMENT,
: 149 0869 1 FDL$$SYNTAX_ERROR,
: 150 0870 1 FDL$$ERROR_CHK,
: 151 0871 1 FDL$$READ_ERROR : NOVALUE;
: 152 0872 1
: 153 0873 1 EXTERNAL
: 154 0874 1
: 155 0875 1 ! Parse control
: 156 0876 1 !
: 157 0877 1 FDL$AB_LINE : DESC_BLK,
: 158 0878 1 FDL$AB_UPCASED : DESC_BLK,
: 159 0879 1 FDL$AB_ITEM : DESC_BLK,
: 160 0880 1 FDL$AB_FDL_STRING : DESC_BLK,
: 161 0881 1 FDL$AB_PRE_PARSE_BLOCK : BLOCK [ ,BYTE ],
: 162 0882 1 FDL$AB_PRE_PARSE_STATE,
: 163 0883 1 FDL$AB_PRE_PARSE_KEY,
: 164 0884 1 FDL$GL_STNOMPTR,
: 165 0885 1 FDL$GL_MAXLINE,
: 166 0886 1 FDL$AB_CTRL : BLOCK [ ,BYTE ],
: 167 0887 1 FDL$GL_PRIMARY,
: 168 0888 1 FDL$GL_PRINUM,
: 169 0889 1 FDL$AB_PRICTRL : BLOCK [ ,BYTE ],
: 170 0890 1 FDL$GL_SECONDARY,
: 171 0891 1 FDL$GL_SECNUM,
: 172 0892 1 FDL$AB_SECCTRL : BITVECTOR [ FDL$K_SCTRL_VEC ],
: 173 0893 1 FDL$AB_SECCTRL : VECTOR [ FDL$K_SCTRL_LONG, LONG ],
: 174 0894 1 FDL$GL_QUALIFIER,
: 175 0895 1 FDL$GL_NUMBER,
: 176 0896 1 FDL$GL_SWITCH,
```



```
: 177      0897 1      FDL$GL_PROTECTION,
: 178      0898 1      FDL$AL_DATE_TIME      : VECTOR [ ,LONG ],
: 179      0899 1      FDL$AB_STRING          : DESC_BLK,
: 180      0900 1      FDL$AB_COMMENT          : DESC_BLK,
: 181      0901 1      FDL$GL_STMNTNUM,
: 182      0902 1      FDL$AB_FDL_RAB          : $RAB_DECL;
: 183      0903 1
: 184      0904 1 LITERAL
: 185      0905 1      SMALL_A                = 97,      ! ASCII character "a"
: 186      0906 1      SMALL_Z                = 122,      ! ASCII character "z"
: 187      0907 1      UPCASE_MASK             = 32,      ! Mask to convert to upercase ASCII
: 188      0908 1      COMMENT_MARK            = 33;      ! ASCII character "!"
: 189      0909 1
: 190      0910 1 OWN
: 191      0911 1      STRING_DESC             : DESC_BLK;
```

```
193 0912 1 %SBTTL 'GET_LINE'
194 0913 1 GLOBAL ROUTINE FDL$$GET_LINE =
195 0914 1 ++
196 0915 1
197 0916 1 Functional Description:
198 0917 1
199 0918 1 Set up a new item for the parse tables. If there are no
200 0919 1 more items on a line it then reads from the input file (or uses
201 0920 1 the FDL STRING)
202 0921 1 It then upcases it; inits some values and returns
203 0922 1
204 0923 1 Calling Sequence:
205 0924 1
206 0925 1 Called from the parse tables
207 0926 1
208 0927 1 Input Parameters:
209 0928 1 none
210 0929 1
211 0930 1 Implicit Inputs:
212 0931 1 none
213 0932 1
214 0933 1 Output Parameters:
215 0934 1 none
216 0935 1
217 0936 1 Implicit Outputs:
218 0937 1 none
219 0938 1
220 0939 1 Routine Value:
221 0940 1 none
222 0941 1
223 0942 1 Side Effects:
224 0943 1 none
225 0944 1
226 0945 1 --
227 0946 1
228 0947 2 BEGIN
229 0948 2
230 0949 2 TPARSE_ARGS;
231 0950 2
232 0951 2 ! Main processing loop
233 0952 2 !
234 0953 2 DO
235 0954 2 BEGIN
236 0955 2
237 0956 2 ! If there are no more items in the line get a new line
238 0957 2 !
239 0958 2 IF .FDL$AB_ITEM [ DSC$W_LENGTH ] EQL 0
240 0959 2 THEN
241 0960 2 BEGIN
242 0961 2
243 0962 2 IF .FDL$AB_CTRL [ FDL$V_STRING_SPEC ]
244 0963 2 THEN
245 0964 2 BEGIN
246 0965 2
247 0966 2 ! Only go thru once for the string.
248 0967 2 ! Don't go thru at all if the string is null.
249 0968 2 !
```



```
250 0969 6 IF (
251 0970 7 (.FDLSAB_CTRL [ FDL$V_USED_STRING ])
252 0971 6 OR
253 0972 7 (.FDLSAB_FDL_STRING [ DSC$W_LENGTH ] EQLU 0)
254 0973 5 ) THEN
255 0974 5 RETURN 0;
256 0975 5
257 0976 5 CH$MOVE ( .FDLSAB_FDL_STRING [ DSC$W_LENGTH ],
258 0977 5 .FDLSAB_FDL_STRING [ DSC$A_POINTER ],
259 0978 5 .FDLSAB_LINE [ DSC$A_POINTER ] );
260 0979 5
261 0980 5 FDL$AB_LINE [ DSC$W_LENGTH ] = .FDLSAB_FDL_STRING [ DSC$W_LENGTH ];
262 0981 5 FDL$AB_CTRL [ FDL$V_USED_STRING ] = _SET;
263 0982 5
264 0983 5 END
265 0984 4 ELSE
266 0985 5 BEGIN
267 0986 5
268 0987 5 ! Loop until we get a non-zero line
269 0988 5 !
270 0989 5 DO
271 0990 6 BEGIN
272 0991 6
273 0992 6 ! Get the new line from the FDL file.
274 0993 6 !
275 0994 6 RET_ON_ERROR( $GET ( RAB=FDLSAB_FDL_RAB,ERR=FDL$$READ_ERROR ) );
276 0995 6
277 0996 6 END
278 0997 6
279 0998 6 UNTIL ( FDL$AB_LINE [ DSC$W_LENGTH ] =
280 0999 5 .FDL$AB_FDL_RAB [ RAB$W_RSZ ] ) NEQ 0;
281 1000 5
282 1001 4 END;
283 1002 4
284 1003 4 ! Up case the whole line and move it into the upcase buffer
285 1004 4 !
286 1005 4 UPCASE();
287 1006 4
288 1007 4 ! Point the tables to the upcased line
289 1008 4 !
290 1009 4 FDL$AB_ITEM [ DSC$A_POINTER ] = .FDLSAB_UPCASED [ DSC$A_POINTER ]
291 1010 4
292 1011 3 END;
293 1012 3
294 1013 3 ! Point to the next item
295 1014 3 !
296 1015 3 FDL$AB_ITEM [ DSC$A_POINTER ] = .FDLSAB_ITEM [ DSC$A_POINTER ] +
297 1016 3 .FDLSAB_ITEM [ DSC$W_LENGTH ];
298 1017 3
299 1018 4 BEGIN
300 1019 4
301 1020 4 ! Get the string
302 1021 4 !
303 1022 4 FDL$AB_PRE_PARSE_BLOCK [ TPASL_STRINGCNT ] =
304 1023 4 .FDLSAB_UPCASED [ DSC$W_LENGTH ]
305 1024 5 - (
306 1025 5 .FDLSAB_ITEM [ DSC$A_POINTER ]
```

```

307      1026 5
308      1027 4
309      1028 4      FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ] = .FDL$AB_ITEM [ DSC$A_POINTER ];
310      1029 4
311      1030 4      ! Find where to chop it off - the Tparse will set these flags if
312      1031 4      ! it finds " or '
313      1032 4
314      1033 4      FDL$AB_CTRL [ FDL$V_QUOTE_PRES ] = _CLEAR;
315      1034 4      FDL$AB_CTRL [ FDL$V_APOST_PRES ] = _CLEAR;
316      1035 4
317      1036 4      RET_ON_ERROR ( LIB$TPARSE (
P 318      1037 4      FDL$AB_PRE_PARSE_BLOCK,FDL$AB_PRE_PARSE_STATE,FDL$AB_PRE_PARSE_KEY ));
319      1038 4
320      1039 4      ! Now set up the main tparse block to find our 'line'
321      1040 4
322      1041 4      TPARSE_BLOCK [ TPA$L_STRINGPTR ] = .FDL$AB_ITEM [ DSC$A_POINTER ];
323      1042 4      TPARSE_BLOCK [ TPA$L_STRINGCNT ] =
324      1043 4      .FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ]
325      1044 4      - .FDL$AB_ITEM [ DSC$A_POINTER ];
326      1045 4
327      1046 4      FDL$AB_ITEM [ DSC$W_LENGTH ] = .TPARSE_BLOCK [ TPA$L_STRINGCNT ];
328      1047 4
329      1048 4      END      ! of local
330      1049 4
331      1050 3      END      ! of main loop
332      1051 3
333      1052 2      UNTIL .FDL$AB_ITEM [ DSC$W_LENGTH ] NEQ 0;
334      1053 2
335      1054 2      FDL$GL_STMNTNUM = .FDL$GL_STMNTNUM + 1;
336      1055 2
337      1056 2      ! Update the user's cell that contains the statement number.
338      1057 2
339      1058 2      IF .FDL$AB_CTRL [ FDL$V_STVALID ]
340      1059 2      THEN
341      1060 2      .FDL$GL_STNUMPTR = .FDL$GL_STMNTNUM;
342      1061 2
343      1062 2      ! Since there is a new secondary for each item clear some flags
344      1063 2
345      1064 2      FDL$GL_SECONDARY      = _CLEAR;
346      1065 2      FDL$GL_SECNUM      = _CLEAR;
347      1066 2      FDL$GL_SWITCH      = _CLEAR;
348      1067 2      FDL$GL_PROTECTION = _CLEAR;
349      1068 2      FDL$AB_STRING [ DSC$W_LENGTH ] = 0;
350      1069 2      FDL$AB_CTRL [ FDL$V_WARNING ] = _CLEAR;
351      1070 2      FDL$AB_CTRL [ FDL$V_COMMENT ] = _CLEAR;
352      1071 2      FDL$AB_CTRL [ FDL$V_LINECMT ] = _CLEAR;
353      1072 2
354      1073 2      RETURN SSS_NORMAL
355      1074 2
356      1075 1      END;
```

```

.TITLE FDLDRIVER VAX-11 FDL Utilities
.IDENT \V04-000\

.PSECT _FDL$OWN,NOEXE, PIC,2
```


00000 STRING_DESC:

.BLKB 8

.EXTRN FDL\$\$GET_VM, FDL\$\$FREE_VM
.EXTRN LIB\$TPARSE, STR\$TRIM
.EXTRN SYSS\$BINTIM, FDL\$ FACILITY
.EXTRN FDL\$ FAO_MAX, FDL\$ ABKW
.EXTRN FDL\$ ABPRIKW, FDL\$ CREATE
.EXTRN FDL\$ CREATED, FDL\$ CREATEDSTM
.EXTRN FDL\$ FDLERROR, FDL\$ ILL_ARG
.EXTRN FDL\$ INSVIRMEM, FDL\$ INVBLK
.EXTRN FDL\$ INVDATIM, FDL\$ MULPRI
.EXTRN FDL\$ MULSEC, FDL\$ NOQUAL
.EXTRN FDL\$ NULLPRI, FDL\$ OPENFDL
.EXTRN FDL\$ OUTORDER, FDL\$ OPENOUT
.EXTRN FDL\$ WRITEERR, FDL\$ READERR
.EXTRN FDL\$ RFLOC, FDL\$ TITLE
.EXTRN FDL\$ SYNTAX, FDL\$ VALPRI
.EXTRN FDL\$ UNQUAKW, FDL\$ UNPRIKW
.EXTRN FDL\$ UNSECKW, FDL\$ WARNING
.EXTRN FDL\$AB_LINE, FDL\$AB_UPCASED
.EXTRN FDL\$AB_ITEM, FDL\$AB_FDL_STRING
.EXTRN FDL\$AB_PRE_PARSE_BLOCK
.EXTRN FDL\$AB_PRE_PARSE_STATE
.EXTRN FDL\$AB_PRE_PARSE_KEY
.EXTRN FDL\$GL_STNOMPTR
.EXTRN FDL\$GL_MAXLINE, FDL\$AB_CTRL
.EXTRN FDL\$GL_PRIMARY, FDL\$GL_PRINUM
.EXTRN FDL\$AB_PRICTRL, FDL\$GL_SECONDARY
.EXTRN FDL\$GL_SECNUM, FDL\$AB_SECCTRL
.EXTRN FDL\$AB_SECCTRL
.EXTRN FDL\$GL_QUALIFIER
.EXTRN FDL\$GL_NUMBER, FDL\$GL_SWITCH
.EXTRN FDL\$GL_PROTECTION
.EXTRN FDL\$AL_DATE_TIME
.EXTRN FDL\$AB_STRING, FDL\$AB_COMMENT
.EXTRN FDL\$GL_STMNTNUM
.EXTRN FDL\$AB_FDL_RAB, SYSS\$GET

.PSECT _FDL\$CODE, NOWRT, SHR, PIC, 2

OFFC 00000

.ENTRY FDL\$\$GET_LINE, Save R2,R3,R4,R5,R6,R7,R8,- ; 0913
R9,R10,RT1
MOVAB FDL\$AB_FDL_STRING, R11
MOVAB FDL\$AB_PRE_PARSE_BLOCK+12, R10
MOVAB FDL\$AB_CTRL, R9
MOVAB FDL\$AB_ITEM, R8
MOVZWL FDL\$AB_ITEM, R7 ; 0958
BNEQ 6\$
BBC #4, FDL\$AB_CTRL+1, 4\$; 0962
BBC #5, FDL\$AB_CTRL+1, 3\$; 0970
BRW 10\$
TSTW FDL\$AB_FDL_STRING ; 0972
BEQL 2\$
MOVZWL FDL\$AB_FDL_STRING, R6 ; 0976
MOVL FDL\$AB_FDL_STRING+4, R1 ; 0977
MOVL FDL\$AB_LINE+4, R0 ; 0978

2B 03 01 01 A9 04 E1 00023 1\$:
05 E1 00028
00EE 31 0002D 2\$:
6B B5 00030 3\$:
F9 13 00032
56 6B 3C 00034
51 04 AB D0 00037
50 00000000G 00 D0 0003B

60	00000000G	61	56	28	00042	MOV C3	R6, (R1), (R0)		
	01	A9	56	B0	00046	MOVW	R6, FDL\$AB_LINE		0980
			20	88	0004D	BISB2	#32, FDL\$AB_CTRL+1		0981
			23	11	00051	BRB	5\$		0962
		00000000V	00	9F	00053	4\$: PUSHAB	FDL\$\$READ_ERROR		0994
		00000000G	00	9F	00059	PUSHAB	FDL\$AB_FDL_RAB		
	00000000G	00	02	FB	0005F	CALLS	#2, SY\$\$GET		
		59	50	E9	00066	BLBC	STATUS, 7\$		
	00000000G	00	00	B0	00069	MOVW	FDL\$AB_FDL_RAB+34, FDL\$AB_LINE		0999
			DD	13	00074	BEQL	4\$		
	00000000V	00	00	FB	00076	5\$: CALLS	#0, UPCASE		1005
	04	A8	00	D0	0007D	MOVL	FDL\$AB_UPCASED+4, FDL\$AB_ITEM+4		1009
		50	68	3C	00085	6\$: MOVZWL	FDL\$AB_ITEM, R0		1016
	04	A8	50	C0	00088	ADDL2	R0, FDL\$AB_ITEM+4		
		51	04	A8	D0	0008C	MOVL	FDL\$AB_ITEM+4, R1	1025
50	00000000G	00	51	C3	00090	SUBL3	R1, FDL\$AB_UPCASED+4, R0		1026
		52	00	3C	00098	MOVZWL	FDL\$AB_UPCASED, R2		1024
FC	AA	50	52	C1	0009F	ADDL3	R2, R0, FDL\$AB_PRE_PARSE_BLOCK+8		
		6A	51	D0	000A4	MOVL	R1, FDL\$AB_PRE_PARSE_BLOCK+12		1028
	01	A9	00	8F	8A	000A7	BICB2	#192, FDL\$AB_CTRL+1	1034
			00	9F	000AC	PUSHAB	FDL\$AB_PRE_PARSE_KEY		1037
		00000000G	00	9F	000B2	PUSHAB	FDL\$AB_PRE_PARSE_STATE		
			00	9F	000B8	PUSHAB	FDL\$AB_PRE_PARSE_BLOCK		
	00000000G	00	03	FB	000BB	CALLS	#3, LIB\$TPARSE		
		5B	50	E9	000C2	7\$: BLBC	STATUS, 11\$		
		50	04	A8	D0	000C5	MOVL	FDL\$AB_ITEM+4, R0	1041
08	AC	OC	50	D0	000C9	MOVL	R0, 12(TPARSE_BLOCK)		
		6A	50	C3	000CD	SUBL3	R0, FDL\$AB_PRE_PARSE_BLOCK+12, -		1044
							8(TPARSE_BLOCK)		
		68	08	AC	B0	000D2	MOVW	8(TPARSE_BLOCK), FDL\$AB_ITEM	1046
		57	68	3C	000D6	MOVZWL	FDL\$AB_ITEM, R7		1052
			03	12	000D9	BNEQ	8\$		
			FF	45	31	000DB	BRW	1\$	
		00000000G	00	D6	000DE	8\$: INCL	FDL\$GL_STMNTNUM		1054
0E	02	A9	02	E1	000E4	BBC	#2, FDL\$AB_CTRL+2, 9\$		1058
		50	00	D0	000E9	MOVL	FDL\$GL_STNMPTR, R0		1060
		60	00	D0	000F0	MOVL	FDL\$GL_STMNTNUM, (R0)		
			00	D4	000F7	9\$: CLRL	FDL\$GL_SECONDARY		1064
			00	D4	000FD	CLRL	FDL\$GL_SECNUM		1065
			00	D4	00103	CLRL	FDL\$GL_SWITCH		1066
			00	D4	00109	CLRL	FDL\$GL_PROTECTION		1067
			00	B4	0010F	CLRW	FDL\$AB_STRING		1068
		69	0308	8F	AA	00115	BICW2	#776, FDL\$AB_CTRL	1071
		50	01	D0	0011A	MOVL	#1, R0		1073
				04	0011D	RET			
			50	D4	0011E	10\$: CLRL	R0		1075
				04	00120	11\$: RET			

; Routine Size: 289 bytes, Routine Base: _FDL\$CODE + 0000


```

: 358      1076 1 %SBTTL 'UPCASE'
: 359      1077 1 ROUTINE UPCASE : NOVALUE =
: 360      1078 1 ++
: 361      1079 1
: 362      1080 1 Functional Description:
: 363      1081 1
: 364      1082 1     Upcases the input line while moving it into the upcase buffer
: 365      1083 1
: 366      1084 1 Calling Sequence:
: 367      1085 1
: 368      1086 1     UPCASE()
: 369      1087 1
: 370      1088 1 Input Parameters:
: 371      1089 1     none
: 372      1090 1
: 373      1091 1 Implicit Inputs:
: 374      1092 1
: 375      1093 1     FDL$AB_LINE      - Descriptor of the input line
: 376      1094 1
: 377      1095 1 Output Parameters:
: 378      1096 1     none
: 379      1097 1
: 380      1098 1 Implicit Outputs:
: 381      1099 1
: 382      1100 1     FDL$AB_UPCASED - Descriptor of the upcased input line
: 383      1101 1
: 384      1102 1 Routine Value:
: 385      1103 1     none
: 386      1104 1
: 387      1105 1 Side Effects:
: 388      1106 1     none
: 389      1107 1
: 390      1108 1 --
: 391      1109 1
: 392      1110 2 BEGIN
: 393      1111 2
: 394      1112 2 LOCAL
: 395      1113 2     CHAR      : REF VECTOR [ ,BYTE ],
: 396      1114 2     UPCR      : REF VECTOR [ ,BYTE ];
: 397      1115 2
: 398      1116 2     ! Point to the string of characters and the upcase buffer
: 399      1117 2
: 400      1118 2     CHAR = .FDL$AB_LINE [ DSC$A_POINTER ];
: 401      1119 2     UPCR = .FDL$AB_UPCASED [ DSC$A_POINTER ];
: 402      1120 2
: 403      1121 2     ! Loop for all the characters in a line
: 404      1122 2
: 405      1123 2     INCR I FROM 0 TO ( .FDL$AB_LINE [ DSC$W_LENGTH ] - 1 ) BY 1
: 406      1124 2     DO
: 407      1125 2
: 408      1126 2         ! If the char. is a lower case letter upcase it
: 409      1127 2         ! else just copy it over
: 410      1128 2
: 411      1129 3         IF ( .CHAR [ .I ] GEQU SMALL_A ) AND ( .CHAR [ .I ] LEQU SMALL_Z )
: 412      1130 3         THEN
: 413      1131 3             UPCR [ .I ] = .CHAR [ .I ] AND ( NOT UPCASE_MASK )
: 414      1132 2         ELSE
```

```
: 415      1133  2      UPCR [ .I ] = .CHAR [ .I ];
: 416      1134  2
: 417      1135  2      ! Set the length of the upcased line
: 418      1136  2
: 419      1137  2      FDL$AB_UPCASED [ DSC$W_LENGTH ] = .FDL$AB_LINE [ DSC$W_LENGTH ];
: 420      1138  2
: 421      1139  2      RETURN
: 422      1140  2
: 423      1141  1      END;
```

			000C 00000	UPCASE: .WORD	Save R2,R3	: 1077
	51	00000000G	00 D0 00002	MOVL	FDL\$AB_LINE+4, CHAR	: 1118
	50	00000000G	00 D0 00009	MOVL	FDL\$AB_UPCASED+4, UPCR	: 1119
	53	00000000G	00 3C 00010	MOVZWL	FDL\$AB_LINE, R3	: 1123
	52		01 CE 00017	MNEGL	#1, I	: 1131
			1B 11 0001A	BRB	3\$	
	61	8F	6241 91 0001C	1\$: CMPB	(I)[CHAR], #97	: 1129
			0F 1F 00021	BLSSU	2\$	
	7A	8F	6241 91 00023	CMPB	(I)[CHAR], #122	
			08 1A 00028	BGTRU	2\$	
	6240	6241	20 8B 0002A	BICB3	#32, (I)[CHAR], (I)[UPCR]	: 1131
			05 11 00030	BRB	3\$	
		6240	6241 90 00032	2\$: MOVB	(I)[CHAR], (I)[UPCR]	: 1133
	E1	52	53 F2 00037	3\$: AOBLS	R3, I, 1\$: 1129
		00000000G 00	53 B0 0003B	MOVW	R3, FDL\$AB_UPCASED	: 1137
			04 00042	RET		: 1141

; Routine Size: 67 bytes, Routine Base: _FDL\$CODE + 0121


```
: 425      1142 1 %SBTTL 'SET LINE'
: 426      1143 1 GLOBAL ROUTINE FDL$$SET_LINE =
: 427      1144 1 ++
: 428      1145 1
: 429      1146 1 Functional Description:
: 430      1147 1
: 431      1148 1 Calling Sequence:
: 432      1149 1
: 433      1150 1 Input Parameters:
: 434      1151 1 none
: 435      1152 1
: 436      1153 1 Implicit Inputs:
: 437      1154 1 none
: 438      1155 1
: 439      1156 1 Output Parameters:
: 440      1157 1 none
: 441      1158 1
: 442      1159 1 Implicit Outputs:
: 443      1160 1 none
: 444      1161 1
: 445      1162 1 Routine Value:
: 446      1163 1 none
: 447      1164 1
: 448      1165 1 Side Effects:
: 449      1166 1 none
: 450      1167 1
: 451      1168 1 --
: 452      1169 1
: 453      1170 2 BEGIN
: 454      1171 2
: 455      1172 2 TPARSE_ARGS;
: 456      1173 2
: 457      1174 2 FDL$AB_ITEM [ DSC$A_POINTER ] = .TPARSE_BLOCK [ TPA$L_TOKENPTR ];
: 458      1175 2
: 459      1176 2 RETURN SSS_NORMAL
: 460      1177 2
: 461      1178 1 END;
```

```
00000000G 00      14  AC 0000 0000
                   50  01 D0 00002
                   04 D0 0000A
                   04 0000D
```

```
.ENTRY FDL$$SET_LINE, Save nothing
MOVL 20(TPARSE_BLOCK), FDL$AB_ITEM+4
MOVL #1, R0
RET
```

```
: 1143
: 1174
: 1176
: 1178
```

; Routine Size: 14 bytes, Routine Base: _FDL\$CODE + 0164

```
: 463      1179 1 %SBTTL 'SET TERM'
: 464      1180 1 GLOBAL ROUTINE FDL$$SET_TERM =
: 465      1181 1 ++
: 466      1182 1
: 467      1183 1 Functional Description:
: 468      1184 1
: 469      1185 1 Calling Sequence:
: 470      1186 1
: 471      1187 1 Input Parameters:
: 472      1188 1 none
: 473      1189 1
: 474      1190 1 Implicit Inputs:
: 475      1191 1 none
: 476      1192 1
: 477      1193 1 Output Parameters:
: 478      1194 1 none
: 479      1195 1
: 480      1196 1 Implicit Outputs:
: 481      1197 1 none
: 482      1198 1
: 483      1199 1 Routine Value:
: 484      1200 1 none
: 485      1201 1
: 486      1202 1 Side Effects:
: 487      1203 1 none
: 488      1204 1
: 489      1205 1 --
: 490      1206 1
: 491      1207 2 BEGIN
: 492      1208 2
: 493      1209 2 TPARSE_ARGS;
: 494      1210 2
: 495      1211 2 FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ] =
: 496      1212 2 .FDL$AB_PRE_PARSE_BLOCK [ TPA$L_STRINGPTR ] - 1;
: 497      1213 2
: 498      1214 2 RETURN SS$_NORMAL
: 499      1215 2
: 500      1216 1 END;
```

```
0000 00000
50 00000000G 00 D7 00002
01 D0 00008
04 0000B
```

```
.ENTRY FDL$$SET_TERM, Save nothing
DECL FDL$AB_PRE_PARSE_BLOCK+12
MOVL #1, R0
RET
```

```
: 1180
: 1212
: 1214
: 1216
```

; Routine Size: 12 bytes, Routine Base: _FDL\$CODE + 0172


```
1217 1 %SBTTL 'SET PRIMARY'
1218 1 GLOBAL ROUTINE FDL$$SET_PRIMARY =
1219 1 ++
1220 1
1221 1 Functional Description:
1222 1
1223 1 Calling Sequence:
1224 1
1225 1 Input Parameters:
1226 1 none
1227 1
1228 1 Implicit Inputs:
1229 1 none
1230 1
1231 1 Output Parameters:
1232 1 none
1233 1
1234 1 Implicit Outputs:
1235 1 none
1236 1
1237 1 Routine Value:
1238 1 none
1239 1
1240 1 Side Effects:
1241 1 none
1242 1
1243 1 --
1244 1
1245 2 BEGIN
1246 2
1247 2 TPARSE_ARGS;
1248 2
1249 2 OWN
1250 2 NXTPRINUM; ! The next key or area primary number
1251 2
1252 2 LOCAL
1253 2 PRIMASK;
1254 2
1255 2 PRIMASK = .TPARSE_BLOCK [ TPA$$_PARAM ];
1256 2
1257 2 ! If this is the first call then clear and go else check to make sure a
1258 2 secondary was processed.
1259 2
1260 2 IF .FDL$AB_CTRL [ FDL$V_INITIAL ]
1261 2 THEN
1262 2 FDL$AB_CTRL [ FDL$V_INITIAL ] = _CLEAR
1263 2 ELSE
1264 2
1265 2 ! If a secondary was processed the ok else null primary warning
1266 2
1267 2 IF .FDL$AB_CTRL [ FDL$V_SECONDARY ]
1268 2 THEN
1269 2 FDL$AB_CTRL [ FDL$V_SECONDARY ] = _CLEAR
1270 2 ELSE
1271 2 SIGNAL ( FDL$_NULLPRI );
1272 2
1273 3 IF (
```

```

559 1274 4 ( NOT .FDLSAB_CTRL [ FDL$V_DFLT_PRES ] )
560 1275 3 OR
561 1276 4 ( .FDLSAB_CTRL [ FDL$V_REPARSE ] )
562 1277 2 ) THEN
563 1278 2 BEGIN
564 1279 2
565 1280 2 ! If this primary has been defied before check to see if it's a
566 1281 2 ! key or area primary
567 1282 2
568 1283 2 IF ( .PRIMASK AND .FDLSAB_PRICTL ) NEQU 0
569 1284 2 THEN
570 1285 2
571 1286 2 ! Is it a key, area, analysis_of_key or analysis_of_area primary
572 1287 2 ! check the order in case the last was the same
573 1288 2
574 1289 4 IF (
575 1290 5 (
576 1291 6 ( .PRIMASK )
577 1292 5 AND
578 1293 6 ( FDL$M_KEY OR FDL$M_AREA OR FDL$M_ANALK OR FDL$M_ANALA )
579 1294 4 ) NEQU 0
580 1295 4
581 1296 3 ) THEN
582 1297 3
583 1298 3 ! What was the last primary
584 1299 3
585 1300 4 IF (
586 1301 5 ( .FDL$GL_PRIMARY EQLU FDL$C_KEY )
587 1302 4 OR
588 1303 5 ( .FDL$GL_PRIMARY EQLU FDL$C_AREA )
589 1304 4 OR
590 1305 5 ( .FDL$GL_PRIMARY EQLU FDL$C_ANALK )
591 1306 4 OR
592 1307 5 ( .FDL$GL_PRIMARY EQLU FDL$C_ANALA )
593 1308 3 ) THEN
594 1309 3
595 1310 3 ! Check to see if the number is correct
596 1311 3
597 1312 3 IF .FDL$GL_PRINUM EQLU .NXTPRINUM
598 1313 3 THEN
599 1314 3 NXTPRINUM = .NXTPRINUM + 1
600 1315 3 ELSE
601 1316 4 BEGIN
602 1317 4 SIGNAL( FDL$_OUTORDER,1,.FDL$GL_STMNTNUM );
603 1318 4 RETURN FDL$_SYNTAX
604 1319 4 END
605 1320 3
606 1321 3 ELSE
607 1322 3 NXTPRINUM = 0
608 1323 3
609 1324 2 ELSE
610 1325 2
611 1326 2 ! Multiple primaries is only a warning
612 1327 2
613 1328 2 SIGNAL( FDL$_MULPRI,1,.FDL$GL_STMNTNUM )
614 1329 2
615 1330 2 ELSE
```



```

616 1331
617 1332      ! Is it a first key or area or ect. primary check the number
618 1333      !
619 1334      IF ( .PRIMASK AND ( FDL$M_KEY OR FDL$M_AREA OR FDL$M_ANALK OR
620 1335                          FDL$M_ANALA ) ) NEQU 0
621 1336      THEN
622 1337          ! If so check to see if the number is correct
623 1338          !
624 1339          IF .FDL$GL_PRINUM EQLU 0
625 1340          THEN
626 1341              NXTPRINUM = 1
627 1342          ELSE
628 1343              BEGIN
629 1344                  SIGNAL( FDL$_OUTORDER,1,.FDL$GL_STMNTNUM );
630 1345                  RETURN FDL$_SYNTAX
631 1346              END;
632 1347
633 1348      END;
634 1349
635 1350      ! Flag it for latter
636 1351      FDL$AB_PRICTRL = .FDL$AB_PRICTRL OR .PRIMASK;
637 1352
638 1353      ! Clear FDL$PRIMARY so that tpase can set it on return
639 1354      FDL$GL_PRIMARY = _CLEAR;
640 1355
641 1356      ! Indicate that a new primary has been found
642 1357      FDL$AB_CTRL [ FDL$V_NEWPRI ] = _SET;
643 1358
644 1359      ! Get ready for a new set of secondaries
645 1360      INCR I FROM 0 TO (FDL$K_SCTRL_LONG-1)
646 1361      DO
647 1362          FDL$AB_SECCTRL [ .I ] = _CLEAR;
648 1363
649 1364      RETURN SSS_NORMAL;
650 1365
651 1366      END;
652 1367
653 1368
654 1369
655 1370
656 1371
```

.PSECT _FDL\$OWN,NOEXE, PIC,2

00008 NXTPRINUM:
.BLKB 4

.PSECT _FDL\$CODE,NOWRT, SHR, PIC,2

```

03FC 00000      .ENTRY FDL$$SET_PRIMARY, Save R2,R3,R4,R5,R6,R7,- ; 1218
59 00000000G 00 9E 00002      R8,R9
58 00000000G 00 9E 00009      MOVAB FDL$GL_STMNTNUM, R9
                        MOVAB FDL$GL_PRINUM, R8
```

	57	00000000G	00	9E	00010	MOVAB	FDL\$GL_PRIMARY, R7	:	
	56	00000000G	00	9E	00017	MOVAB	FDL\$AB_PRICTRL, R6	:	
	55	00000000G	00	9E	0001E	MOVAB	LIB\$SIGNAL, R5	:	
	54	000000000	00	9E	00025	MOVAB	NXTPRINUM, R4	:	
	53	00000000G	00	9E	0002C	MOVAB	FDL\$AB_CTRL, R3	:	
	52	20	AC	D0	00033	MOVL	32(TPARSE BLOCK), PRIMASK	:	1255
			63	95	00037	TSTB	FDL\$AB_CTRL	:	1260
			06	18	00039	BGEQ	1\$:	
	63	80	8F	8A	0003B	BICB2	#128, FDL\$AB_CTRL	:	1262
			13	11	0003F	BRB	3\$:	
06	63		06	E1	00041	1\$: BBC	#6, FDL\$AB_CTRL, 2\$:	1267
	63	40	8F	8A	00045	BICB2	#64, FDL\$AB_CTRL	:	1269
			09	11	00049	BRB	3\$:	
		00000000G	8F	DD	0004B	2\$: PUSHL	#FDL\$ NULLPRI	:	1271
	65		01	FB	00051	CALLS	#1, LIB\$SIGNAL	:	
04	A3		01	E1	00054	3\$: BBC	#1, FDL\$AB_CTRL+2, 4\$:	1274
	67	02	A3	E9	00059	4\$: BLBC	FDL\$AB_CTRL+2, 11\$:	1276
			50	D4	0005D	4\$: CLRL	R0	:	1293
041C	8F		52	B3	0005F	BITW	PRIMASK, #1052	:	
			02	13	00064	BEQL	5\$:	
			50	D6	00066	INCL	R0	:	
	66		52	D3	00068	5\$: BITL	PRIMASK, FDL\$AB_PRICTRL	:	1283
			36	13	0006B	BEQL	9\$:	
	24		50	E9	0006D	BLBC	R0, 8\$:	1289
	50		67	D0	00070	MOVL	FDL\$GL_PRIMARY, R0	:	1301
	0B		50	D1	00073	CMPL	R0, #1T	:	
			0F	13	00076	BEQL	6\$:	
	05		50	D1	00078	CMPL	R0, #5	:	1303
			0A	13	0007B	BEQL	6\$:	
	04		50	D1	0007D	CMPL	R0, #4	:	1305
			05	13	00080	BEQL	6\$:	
	03		50	D1	00082	CMPL	R0, #3	:	1307
			09	12	00085	BNEQ	7\$:	
	64		68	D1	00087	6\$: CMPL	FDL\$GL_PRINUM, NXTPRINUM	:	1312
			23	12	0008A	BNEQ	10\$:	
			64	D6	0008C	INCL	NXTPRINUM	:	1314
			34	11	0008E	BRB	11\$:	
			64	D4	00090	7\$: CLRL	NXTPRINUM	:	1322
			30	11	00092	BRB	11\$:	1300
			69	DD	00094	8\$: PUSHL	FDL\$GL_STMNTNUM	:	1328
			01	DD	00096	PUSHL	#1	:	
		00000000G	8F	DD	00098	PUSHL	#FDL\$ MULPRI	:	
65			03	FB	0009E	CALLS	#3, LIB\$SIGNAL	:	
			21	11	000A1	BRB	11\$:	1289
	1E		50	E9	000A3	9\$: BLBC	R0, 11\$:	1335
			68	D5	000A6	TSTL	FDL\$GL_PRINUM	:	1340
			05	12	000A8	BNEQ	10\$:	
	64		01	D0	000AA	MOVL	#1, NXTPRINUM	:	1342
			15	11	000AD	BRB	11\$:	
			69	DD	000AF	10\$: PUSHL	FDL\$GL_STMNTNUM	:	1345
			01	DD	000B1	PUSHL	#1	:	
		00000000G	8F	DD	000B3	PUSHL	#FDL\$ OUTORDER	:	
65			03	FB	000B9	CALLS	#3, LIB\$SIGNAL	:	
	50	00000000G	8F	D0	000BC	MOVL	#FDL\$_SYNTAX, R0	:	1346
			04	000C3	RET			:	
	66		52	C8	000C4	11\$: BISL2	PRIMASK, FDL\$AB_PRICTRL	:	1353
			67	D4	000C7	CLRL	FDL\$GL_PRIMARY	:	1357

FDLDRIVER
V04-000

VAX-11 FDL Utilities
SET_PRIMARY

D 10
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 19
(8)

63	20	88	000C9	BISB2	#32, FDL\$AB_CTRL
	50	D4	000CC	CLRL	I
	00000000G0040	D4	000CE	CLRL	FDL\$AB_SECCTRL[I]
F5	50	05	F3 000D5	AOBLEQ	#5, I, 12\$
	50	01	D0 000D9	MOVL	#1, R0
		04	000DC	RET	

: 1361
: 1365
: 1367
: 1369
: 1371

; Routine Size: 221 bytes, Routine Base: _FDL\$CODE + 017E

```
: 658      1372 1 %SBTTL 'SET SECONDARY'
: 659      1373 1 GLOBAL ROUTINE FDL$$SET_SECONDARY =
: 660      1374 1 ++
: 661      1375 1
: 662      1376 1 Functional Description:
: 663      1377 1
: 664      1378 1 Calling Sequence:
: 665      1379 1
: 666      1380 1 Input Parameters:
: 667      1381 1 none
: 668      1382 1
: 669      1383 1 implicit Inputs:
: 670      1384 1 none
: 671      1385 1
: 672      1386 1 Output Parameters:
: 673      1387 1 none
: 674      1388 1
: 675      1389 1 Implicit Outputs:
: 676      1390 1 none
: 677      1391 1
: 678      1392 1 Routine Value:
: 679      1393 1 none
: 680      1394 1
: 681      1395 1 Side Effects:
: 682      1396 1 none
: 683      1397 1
: 684      1398 1 --
: 685      1399 1
: 686      1400 2 BEGIN
: 687      1401 2
: 688      1402 2 TPARSE_ARGS;
: 689      1403 2
: 690      1404 2 LOCAL
: 691      1405 2 SECBIT : LONG;
: 692      1406 2
: 693      1407 2 SECBIT = .TPARSE_BLOCK [ TPA$$_PARAM ];
: 694      1408 2
: 695      1409 2 ! See if the secondary has been defined before
: 696      1410 2
: 697      1411 2 IF .FDL$AB_SECCTRL [ .SECBIT ]
: 698      1412 2 THEN
: 699      1413 2
: 700      1414 2 ! If it has then see if it was a key segment thing
: 701      1415 2
: 702      1416 3 IF (
: 703      1417 4 ( .SECBIT EQLU FDL$$_SEGPOS )
: 704      1418 3 OR
: 705      1419 4 ( .SECBIT EQLU FDL$$_SEGLen )
: 706      1420 3 OR
: 707      1421 4 ( .SECBIT EQLU FDL$$_SEGTYP )
: 708      1422 2 ) THEN
: 709      1423 3 BEGIN
: 710      1424 3
: 711      1425 3 ! If it's out of bounds it's an error
: 712      1426 3
: 713      1427 3 IF .FDL$$_GL_SECNUM GTR 7
: 714      1428 3 THEN
```



```
: 715      1429  4      BEGIN
: 716      1430  4      SIGNAL( FDL$ UNSECKW,3,
: 717      1431  4          .FDL$GL_STMNTNUM,
: 718      1432  4          .TPARSE_BLOCK [ TPA$L_TOKENCNT ],
: 719      1433  4          .TPARSE_BLOCK [ TPA$L_TOKENPTR ] );
: 720      1434  4      RETURN FDL$_SYNTAX
: 721      1435  4      END
: 722      1436  3      END
: 723      1437  2      ELSE
: 724      1438  2          ! If it has been defined before it's only a warning
: 725      1439  2          !
: 726      1440  2          SIGNAL( FDL$_MULSEC,1,.FDL$GL_STMNTNUM )
: 727      1441  2      ELSE
: 728      1442  2          ! Flag it for next time (unless it's an ACL ENTRY - which can be repeated)
: 729      1443  2          !
: 730      1444  2          IF .SECBIT NEQU FDL$_ACE
: 731      1445  2          THEN
: 732      1446  2              FDL$AB_SECCTRL [ .SECBIT ] = _SET;
: 733      1447  2          !
: 734      1448  2          ! Get ready for a new an wonderful qualifier
: 735      1449  2          !
: 736      1450  2          FDL$GL_QUALIFIER = _CLEAR;
: 737      1451  2      RETURN SS$_NORMAL
: 738      1452  2      END;
: 739      1453  2
: 740      1454  2
: 741      1455  2
: 742      1456  1
```

			001C 00000	.ENTRY	FDL\$\$SET SECONDARY, Save R2,R3,R4	: 1373
	54	00000000G	00 9E 00002	MOVAB	FDL\$AB_SECCTRL, R4	
	53	00000000G	00 9E 00009	MOVAB	LIB\$SIGNAL, R3	
	52	00000000G	00 9E 00010	MOVAB	FDL\$GL_STMNTNUM, R2	
	50	20	AC D0 00017	MOVL	32(TPARSE_BLOCK), SECBIT	: 1407
4C	64		50 E1 0001B	BBC	SECBIT, FDL\$AB_SECCTRL, 3\$: 1411
00000086	8F		50 D1 0001F	CMPL	SECBIT, #134	: 1417
			12 13 00026	BEQL	1\$	
00000085	8F		50 D1 00028	CMPL	SECBIT, #133	: 1419
			09 13 0002F	BEQL	1\$	
00000087	8F		50 D1 00031	CMPL	SECBIT, #135	: 1421
			22 12 00038	BNEQ	2\$	
	07	00000000G	00 D1 0003A 1\$:	CMPL	FDL\$GL_SECNUM, #7	: 1427
	7E	10	AC 7D 00043	BLEQ	4\$	
			62 DD 00047	MOVQ	16(TPARSE_BLOCK), -(SP)	: 1432
			03 DD 00049	PUSHL	FDL\$GL_STMNTNUM	: 1431
			8F DD 0004B	PUSHL	#3	: 1430
		00000000G	8F DD 0004B	PUSHL	#FDL\$ UNSECKW	
63			05 FB 00051	CALLS	#5, LIB\$SIGNAL	
50		00000000G	8F D0 00054	MOVL	#FDL\$_SYNTAX, R0	: 1434
			04 0005B	RET		
			62 DD 0005C 2\$:	PUSHL	FDL\$GL_STMNTNUM	: 1441
			01 DD 0005E	PUSHL	#1	
		00000000G	8F DD 00060	PUSHL	#FDL\$_MULSEC	

FDLDRIVER
V04-000

VAX-11 FDL Utilities
SET_SECONDARY

G 10
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 22
(9)

63	03	FB	00066	CALLS	#3, LIB\$SIGNAL	:	1416	
08	09	11	00069	BRB	4\$:	1446	
	50	D1	0006B	3\$:	CMP	SECBIT, #8	:	1448
	04	13	0006E	BEQ	4\$:	1452	
00	64	50	E2 00070	BBSS	SECBIT, FDL\$AB_SECCTRL, 4\$:	1454	
		00	D4 00074	4\$:	CLRL	FDL\$GL_QUALIFIER	:	1456
50		01	D0 0007A	MOVL	#1, R0	:		
		04	0007D	RET		:		

; Routine Size: 126 bytes, Routine Base: _FDL\$CODE + 025B


```
: 744      1457 1 %SBTTL 'START_STR'
: 745      1458 1 GLOBAL ROUTINE FDL$$START_STR =
: 746      1459 1 ++
: 747      1460 1
: 748      1461 1   Functional Description:
: 749      1462 1
: 750      1463 1       Initializes the string descriptor
: 751      1464 1
: 752      1465 1   Calling Sequence:
: 753      1466 1
: 754      1467 1       Called from the parse tables
: 755      1468 1
: 756      1469 1   Input Parameters:
: 757      1470 1       none
: 758      1471 1
: 759      1472 1   Implicit Inputs:
: 760      1473 1       none
: 761      1474 1
: 762      1475 1   Output Parameters:
: 763      1476 1       none
: 764      1477 1
: 765      1478 1   Implicit Outputs:
: 766      1479 1       none
: 767      1480 1
: 768      1481 1   Routine Value:
: 769      1482 1       none
: 770      1483 1
: 771      1484 1   Side Effects:
: 772      1485 1       none
: 773      1486 1
: 774      1487 1 --
: 775      1488 1
: 776      1489 2   BEGIN
: 777      1490 2
: 778      1491 2   TPARSE_ARGS;
: 779      1492 2
: 780      1493 2   ! Start the makings of a descriptor
: 781      1494 2   !
: 782      1495 2   FDL$AB_STRING [ DSC$A_POINTER ] = .TPARSE_BLOCK [ TPA$L_TOKENPTR ];
: 783      1496 2
: 784      1497 2   ! Process blanks
: 785      1498 2   !
: 786      1499 2   TPARSE_BLOCK [ TPA$V_BLANKS ] = _SET;
: 787      1500 2
: 788      1501 2   RETURN SS$_NORMAL
: 789      1502 2
: 790      1503 1   END;
```

```
00000000G 00      14      0000 00000
          04 AC      01 DO 00002
          50      01 88 0000A
          01 DO 0000E
          04 00011
```

```
.ENTRY FDL$$START_STR, Save nothing
MOVL 20(TPARSE_BLOCK), FDL$AB_STRING+4
BISB2 #1, 4(TPARSE_BLOCK)
MOVL #1, R0
RET
```

```
: 1458
: 1495
: 1499
: 1501
: 1503
```

FDLDRIVER
V04-000

VAX-11 FDL Utilities
START_STR

I 10
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 24
(10)

; Routine Size: 18 bytes, Routine Base: _FDL\$CODE + 02D9


```

792 1504 1 %SBTTL 'END STR'
793 1505 1 GLOBAL ROUTINE FDL$END_STR =
794 1506 1 ++
795 1507 1
796 1508 1 Functional Description:
797 1509 1
798 1510 1 Terminates the processing of a string and determines the length
799 1511 1
800 1512 1 Calling Sequence:
801 1513 1
802 1514 1 Called from the parse tables
803 1515 1
804 1516 1 Input Parameters:
805 1517 1 none
806 1518 1
807 1519 1 Implicit Inputs:
808 1520 1 none
809 1521 1
810 1522 1 Output Parameters:
811 1523 1 none
812 1524 1
813 1525 1 Implicit Outputs:
814 1526 1 none
815 1527 1
816 1528 1 Routine Value:
817 1529 1 none
818 1530 1
819 1531 1 Side Effects:
820 1532 1 none
821 1533 1
822 1534 1 --
823 1535 1
824 1536 2 BEGIN
825 1537 2
826 1538 2 LOCAL
827 1539 2 SAVE_LEN : WORD,
828 1540 2 CUT_LEN : WORD;
829 1541 2
830 1542 2 TPARSE_ARGS;
831 1543 2
832 1544 2 TPARSE_BLOCK [ TPA$V_BLANKS ] = _CLEAR;
833 1545 2
834 1546 2 ! The size is from where we are minus from where we is
835 1547 2 !
836 1548 2 FDL$AB_STRING [ DSC$W_LENGTH ] = .TPARSE_BLOCK [ TPA$L_STRINGPTR ] -
837 1549 2 .FDL$AB_STRING [ DSC$A_POINTER ];
838 1550 2
839 1551 2 ! If the last char was a "!" then subtract one
840 1552 2 !
841 1553 2 IF .TPARSE_BLOCK [ TPA$B_CHAR ] EQL COMMENT_MARK
842 1554 2 THEN
843 1555 2 FDL$AB_STRING [ DSC$W_LENGTH ] = .FDL$AB_STRING [ DSC$W_LENGTH ] - 1;
844 1556 2
845 1557 2 ! Save this length
846 1558 2 !
847 1559 2 SAVE_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
848 1560 2
```

```
: 849      1561 2      ! Remove trailing blanks
: 850      1562 2
: 851      1563 2      STR$TRIM ( FDL$AB_STRING,FDL$AB_STRING,CUT_LEN );
: 852      1564 2
: 853      1565 2      ! Set the trimmed length
: 854      1566 2
: 855      1567 2      FDL$AB_STRING [ DSC$W_LENGTH ] = .CUT_LEN;
: 856      1568 2
: 857      1569 2      ! Remove any leading white space from the string
: 858      1570 2
: 859      1571 2      FDL$AB_STRING [ DSC$W_LENGTH ] = TRIM_LEADING ();
: 860      1572 2
: 861      1573 2      ! Remove any quotes from the upcased string
: 862      1574 2
: 863      1575 2      FDL$AB_STRING [ DSC$W_LENGTH ] = EXTRACT_QUOTE ();
: 864      1576 2
: 865      1577 2      ! Adjust the pointer so that we are looking into the original input line
: 866      1578 2
: 867      1579 2      FDL$AB_STRING [ DSC$A_POINTER ] = .FDL$AB_STRING [ DSC$A_POINTER ] -
: 868      1580 2                                     .FDL$GL_MAXLINE;
: 869      1581 2
: 870      1582 2      ! Restore the original length
: 871      1583 2
: 872      1584 2      FDL$AB_STRING [ DSC$W_LENGTH ] = .SAVE_LEN;
: 873      1585 2
: 874      1586 2      ! Remove trailing blanks
: 875      1587 2
: 876      1588 2      STR$TRIM ( FDL$AB_STRING,FDL$AB_STRING,CUT_LEN );
: 877      1589 2
: 878      1590 2      ! Set the trimmed length
: 879      1591 2
: 880      1592 2      FDL$AB_STRING [ DSC$W_LENGTH ] = .CUT_LEN;
: 881      1593 2
: 882      1594 2      ! Remove any leading white space from the string
: 883      1595 2
: 884      1596 2      FDL$AB_STRING [ DSC$W_LENGTH ] = TRIM_LEADING ();
: 885      1597 2
: 886      1598 2      ! Remove any quotes from the original string
: 887      1599 2
: 888      1600 2      FDL$AB_STRING [ DSC$W_LENGTH ] = EXTRACT_QUOTE ();
: 889      1601 2
: 890      1602 2      RETURN SS$_NORMAL;
: 891      1603 2
: 892      1604 1      END;
```

				007C 00000	.ENTRY	FDL\$END_STR, Save R2,R3,R4,R5,R6	: 1505
		56	00000000V	00 9E 00002	MOVAB	EXTRACT_QUOTE, R6	
		55	00000000V	00 9E 00009	MOVAB	TRIM_LEADING, R5	
		54	00000000G	00 9E 00010	MOVAB	STR\$TRIM, R4	
		53	00000000G	00 9E 00017	MOVAB	FDL\$AB_STRING, R3	
		5E		04 C2 0001E	SUBL2	#4, SP	
	04	AC		01 8A 00021	BICB2	#1, 4(TPARSE_BLOCK)	: 1544
63	0C	AC	04	A3 A3 00025	SUBW3	FDL\$AB_STRING+4, 12(TPARSE_BLOCK), -	: 1549

21	18	AC	91	0002B	CMPB	FDLSAB_STRING	:	1553
		02	12	0002F	BNEQ	24(TPARSE_BLOCK), #33	:	
		63	B7	00031	DECW	1\$:	1555
52		63	B0	00033	MOVW	FDLSAB_STRING	:	1559
	4008	8F	BB	00036	PUSHR	FDLSAB_STRING, SAVE_LEN	:	1563
		53	DD	0003A	PUSHR	#^M<R3,SP>	:	
64		03	FB	0003C	PUSHL	R3	:	
63		6E	B0	0003F	CALLS	#3, STRSTRIM	:	1567
65		00	FB	00042	MOVW	CUT_LEN, FDL\$AB_STRING	:	1571
63		50	B0	00045	CALLS	#0, -TRIM_LEADING	:	
66		00	FB	00048	MOVW	R0, FDL\$AB_STRING	:	1575
63		50	B0	0004B	CALLS	#0, EXTRACT_QUOTE	:	
04	00000000G	00	C2	0004E	MOVW	R0, FDL\$AB_STRING	:	1580
63		52	B0	00056	SUBL2	FDL\$GL_MAXLINE, FDL\$AB_STRING+4	:	1584
	4008	8F	BB	00059	MOVW	SAVE_LEN, FDL\$AB_STRING	:	1588
		53	DD	0005D	PUSHR	#^M<R3,SP>	:	
64		03	FB	0005F	PUSHL	R3	:	
63		6E	B0	00062	CALLS	#3, STRSTRIM	:	1592
65		00	FB	00065	MOVW	CUT_LEN, FDL\$AB_STRING	:	1596
63		50	B0	00068	CALLS	#0, -TRIM_LEADING	:	
66		00	FB	0006B	MOVW	R0, FDL\$AB_STRING	:	1600
63		50	B0	0006E	CALLS	#0, EXTRACT_QUOTE	:	
50		01	D0	00071	MOVW	R0, FDL\$AB_STRING	:	1602
		04	00074	RETL	#1, R0		:	1604

; Routine Size: 117 bytes, Routine Base: _FDL\$CODE + 02EB

```

: 894      1605 1 %SBTTL 'EXTRACT_QUOTE'
: 895      1606 1 ROUTINE EXTRACT_QUOTE =
: 896      1607 1 ++
: 897      1608 1
: 898      1609 1 Functional Description:
: 899      1610 1
: 900      1611 1     It also extracts out embedded or bracketing quotes or apostrophes
: 901      1612 1
: 902      1613 1 Calling Sequence:
: 903      1614 1
: 904      1615 1     Called from END_STR
: 905      1616 1
: 906      1617 1 Input Parameters:
: 907      1618 1     none
: 908      1619 1
: 909      1620 1 Implicit Inputs:
: 910      1621 1     none
: 911      1622 1
: 912      1623 1 Output Parameters:
: 913      1624 1     none
: 914      1625 1
: 915      1626 1 Implicit Outputs:
: 916      1627 1     none
: 917      1628 1
: 918      1629 1 Routine Value:
: 919      1630 1     The new string length - after the quotes are removed.
: 920      1631 1
: 921      1632 1 Side Effects:
: 922      1633 1     none
: 923      1634 1
: 924      1635 1 --
: 925      1636 1
: 926      1637 2 BEGIN
: 927      1638 2
: 928      1639 2 LOCAL
: 929      1640 2     QCHAR   : BYTE,
: 930      1641 2     J       : LONG,
: 931      1642 2     NEW_LEN : LONG,
: 932      1643 2     CUT_LEN : LONG,
: 933      1644 2     STR     : REF VECTOR [ ,BYTE ],
: 934      1645 2     TMP_STR : REF VECTOR [ ,BYTE ];
: 935      1646 2
: 936      1647 2 NEW_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
: 937      1648 2
: 938      1649 2 ! Now extract out any bracketing or embedded quotes or apostrophes
: 939      1650 2
: 940      1651 2 IF .FDL$AB_CTRL [ FDL$V_QUOTE_PRES ] OR .FDL$AB_CTRL [ FDL$V_APOST_PRES ]
: 941      1652 2 THEN
: 942      1653 2     BEGIN
: 943      1654 2
: 944      1655 2     CUT_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
: 945      1656 2     TMP_STR = FDL$$GET_VM ( .CUT_LEN );
: 946      1657 2
: 947      1658 2     STR = .FDL$AB_STRING [ DSC$A_POINTER ];
: 948      1659 2
: 949      1660 2     IF .FDL$AB_CTRL [ FDL$V_QUOTE_PRES ]
: 950      1661 2     THEN
```



```
: 951      1662  3      QCHAR = ''
: 952      1663  3      ELSE IF .FDLSAB_CTRL [ FDL$V_APOST_PRE ]
: 953      1664  3      THEN
: 954      1665  3          QCHAR = ''';
: 955      1666  3
: 956      1667  3      CH$MOVE ( .CUT_LEN,.FDLSAB_STRING [ DSC$A_POINTER ],.TMP_STR );
: 957      1668  3
: 958      1669  3      NEW_LEN = 0;
: 959      1670  3      J = 0;
: 960      1671  3
: 961      1672  4      WHILE .J LEQ (.CUT_LEN - 1)
: 962      1673  3      DO
: 963      1674  4          BEGIN
: 964      1675  4              ! Now copy the string back, but stripping the QCHARs
: 965      1676  4              ! according to the rules that embedded "" ==> '' and '' ==> '
: 966      1677  4
: 967      1678  4              IF .TMP_STR [ .J ] EQLU .QCHAR
: 968      1679  4              THEN
: 969      1680  4                  BEGIN
: 970      1681  5                      ! If we're not at the beginning or end of the string,
: 971      1682  5                      ! copy one qchar and skip the next
: 972      1683  5                      !
: 973      1684  5                      IF NOT ((.J EQLU 0) OR (.J EQLU (.CUT_LEN-1)))
: 974      1685  5                      THEN
: 975      1686  6                          BEGIN
: 976      1687  6                              IF .TMP_STR [ .J+1 ] EQLU .QCHAR
: 977      1688  6                              THEN
: 978      1689  6                                  J = .J + 1;
: 979      1690  6
: 980      1691  6                                  STR [ .NEW_LEN ] = .TMP_STR [ .J ];
: 981      1692  6                                  NEW_LEN = .NEW_LEN + 1
: 982      1693  6
: 983      1694  6                                  END;
: 984      1695  6
: 985      1696  6                              END;
: 986      1697  5
: 987      1698  5                      END
: 988      1699  5
: 989      1700  4              ELSE
: 990      1701  4                  ! Just copy the character back and bump the count
: 991      1702  4                  !
: 992      1703  5                  BEGIN
: 993      1704  5                      STR [ .NEW_LEN ] = .TMP_STR [ .J ];
: 994      1705  5                      NEW_LEN = .NEW_LEN + 1
: 995      1706  5
: 996      1707  4                  END;
: 997      1708  4                  J = .J + 1;
: 998      1709  4
: 999      1710  3              END;          ! do
: 1000     1711  3
: 1001     1712  3          ! Release the tmp string
: 1002     1713  3          !
: 1003     1714  3          FDL$$FREE_VM ( .CUT_LEN,.TMP_STR );
: 1004     1715  3
: 1005     1716  2      END;          ! IF QUOTE OR APOST PRESENT
: 1006     1717  2
: 1007     1718  2      ! The routine value is the new length
```

```
: 1008
: 1009
: 1010
: 1011
```

```
1719 2 !
1720 2 RETURN .NEW_LEN;
1721 2
1722 1 END;
```

```
OFFC 00000 EXTRACT_QUOTE:
5B 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 1606
50 00000000G 00 3C 00009 MOVAB FDL$AB_CTRL, R11 : 1647
5A 01 50 D0 00010 MOVZWL FDL$AB_STRING, R0 : 1651
01 AB 95 00013 TSTB FDL$AB_CTRL+1 : 1655
05 19 00016 BLSS 1$ : 1656
67 01 AB 06 E1 00018 BBC #6, FDL$AB_CTRL+1, 8$ : 1658
56 50 D0 0001D 1$: MOVL R0, CUT_LEN : 1660
00000000G 00 56 DD 00020 PUSHL CUT_LEN : 1662
57 01 FB 00022 CALLS #1, FDL$$GET_VM : 1663
50 00000000G 00 D0 00029 MOVL R0, TMP_STR : 1665
58 00 D0 0002C MOVL FDL$AB_STRING+4, R0 : 1667
01 50 D0 00033 MOVL R0, STR : 1669
AB 95 00036 TSTB FDL$AB_CTRL+1 : 1670
05 18 00039 BGEQ 2$ : 1672
59 22 90 0003B MOVB #34, QCHAR : 1679
08 11 0003E BRB 3$ : 1686
03 01 AB 06 E1 00040 2$: BBC #6, FDL$AB_CTRL+1, 3$ : 1690
59 27 90 00045 MOVB #39, QCHAR : 1692
67 60 56 28 00048 3$: MOVC3 CUT_LEN, (R0), (TMP_STR) : 1704
5A D4 0004C CLRL NEW_LEN : 1708
50 D4 0004E CLRL J : 1672
51 FF A6 9E 00050 MOVAB -1(R6), R1 : 1679
51 50 D1 00054 4$: CMPL J, R1 : 1686
21 14 00057 BGTR 7$ : 1690
59 6047 91 00059 CMPB (J)[TMP_STR], QCHAR : 1692
12 12 0005D BNEQ 5$ : 1704
50 D5 0005F TSTL J : 1708
13 13 00061 BEQL 6$ : 1672
51 50 D1 00063 CMPL J, R1 : 1714
0E 13 00066 BEQL 6$ : 1720
59 01 A047 91 00068 CMPB 1(J)[TMP_STR], QCHAR : 1722
02 12 0006D BNEQ 5$ :
50 D6 0006F INCL J :
8A48 6047 90 00071 5$: MOVB (J)[TMP_STR], (NEW_LEN)+[STR] :
50 D6 00076 6$: INCL J :
DA 11 00078 BRB 4$ :
7E 56 7D 0007A 7$: MOVQ CUT_LEN, -(SP) :
00000000G 00 02 FB 0007D CALLS #2, FDL$$FREE_VM :
50 5A D0 00084 8$: MOVL NEW_LEN, R0 :
04 00087 RET
```

; Routine Size: 136 bytes, Routine Base: _FDL\$CODE + 0360


```
: 1013 1723 1 %SBTTL 'TRIM_LEADING'
: 1014 1724 1 ROUTINE TRIM_LEADING =
: 1015 1725 1 ++
: 1016 1726 1
: 1017 1727 1 Functional Description:
: 1018 1728 1
: 1019 1729 1 It removes leading spaces and tabs from the input string
: 1020 1730 1
: 1021 1731 1 Calling Sequence:
: 1022 1732 1
: 1023 1733 1 Called from END_STR
: 1024 1734 1
: 1025 1735 1 Input Parameters:
: 1026 1736 1 none
: 1027 1737 1
: 1028 1738 1 Implicit Inputs:
: 1029 1739 1 none
: 1030 1740 1
: 1031 1741 1 Output Parameters:
: 1032 1742 1 none
: 1033 1743 1
: 1034 1744 1 Implicit Outputs:
: 1035 1745 1 none
: 1036 1746 1
: 1037 1747 1 Routine Value:
: 1038 1748 1 The new string length - after the white space is removed.
: 1039 1749 1
: 1040 1750 1 Side Effects:
: 1041 1751 1 none
: 1042 1752 1
: 1043 1753 1 --
: 1044 1754 1
: 1045 1755 2 BEGIN
: 1046 1756 2
: 1047 1757 2 LOCAL
: 1048 1758 2 FLAG : BYTE,
: 1049 1759 2 TMP : BYTE,
: 1050 1760 2 BLANK : BYTE,
: 1051 1761 2 TAB : BYTE,
: 1052 1762 2 J : LONG,
: 1053 1763 2 NEW_LEN : LONG,
: 1054 1764 2 CUT_LEN : LONG,
: 1055 1765 2 STR : REF VECTOR [ ,BYTE ],
: 1056 1766 2 TMP_STR : REF VECTOR [ ,BYTE ];
: 1057 1767 2
: 1058 1768 2 BLANK = ' ';
: 1059 1769 2 TAB = ' ';
: 1060 1770 2 TMP = ..FDL$AB_STRING [ DSC$A_POINTER ];
: 1061 1771 2
: 1062 1772 2 NEW_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
: 1063 1773 2
: 1064 1774 2 ! Now extract out any bracketing or embedded quotes or apostrophes
: 1065 1775 2 !
: 1066 1776 3 IF (.TMP EQLU .BLANK) OR (.TMP EQLU .TAB)
: 1067 1777 2 THEN
: 1068 1778 3 BEGIN
: 1069 1779 3
```

```

: 1070      1780      3      CUT_LEN = .FDL$AB_STRING [ DSC$W_LENGTH ];
: 1071      1781      3      TMP_STR = FDL$$GET_VM ( .CUT_LEN );
: 1072      1782      3
: 1073      1783      3      STR = .FDL$AB_STRING [ DSC$A_POINTER ];
: 1074      1784      3
: 1075      1785      3      CH$MOVE ( .CUT_LEN,.FDL$AB_STRING [ DSC$A_POINTER ],.TMP_STR );
: 1076      1786      3
: 1077      1787      3      NEW_LEN = 0;
: 1078      1788      3      J = 0;
: 1079      1789      3      FLAG = _CLEAR;
: 1080      1790      3
: 1081      1791      4      WHILE .J LEQ (.CUT_LEN - 1)
: 1082      1792      3      DO
: 1083      1793      4      BEGIN
: 1084      1794      4          ! Now copy the string back, but stripping the white space
: 1085      1795      4          !
: 1086      1796      4          IF (.TMP_STR [ .J ] EQLU .BLANK) OR (.TMP_STR [ .J ] EQLU .TAB)
: 1087      1797      5          THEN
: 1088      1798      4              BEGIN
: 1089      1799      5                  ! If we have seen the a non-white character
: 1090      1800      5                  ! just copy this blank or tab like any other char
: 1091      1801      5                  !
: 1092      1802      5                  !
: 1093      1803      5                  IF .FLAG
: 1094      1804      5                  THEN
: 1095      1805      5                      BEGIN
: 1096      1806      6                          STR [ .NEW_LEN ] = .TMP_STR [ .J ];
: 1097      1807      6                          NEW_LEN = .NEW_LEN + 1
: 1098      1808      6                      END;
: 1099      1809      6                  ELSE
: 1100      1810      6                      ! Just copy the character back and bump the count
: 1101      1811      5                      BEGIN
: 1102      1812      5                          FLAG = SET;
: 1103      1813      4                          STR [ .NEW_LEN ] = .TMP_STR [ .J ];
: 1104      1814      4                          NEW_LEN = .NEW_LEN + 1
: 1105      1815      4                      END;
: 1106      1816      5                      J = .J + 1;
: 1107      1817      5                  END;
: 1108      1818      5                  ! do
: 1109      1819      5                  ! Release the tmp string
: 1110      1820      5                  FDL$$FREE_VM ( .CUT_LEN,.TMP_STR );
: 1111      1821      5                  ! The routine value is the new length
: 1112      1822      4                  RETURN .NEW_LEN;
: 1113      1823      4
: 1114      1824      4
: 1115      1825      4
: 1116      1826      3
: 1117      1827      3
: 1118      1828      3
: 1119      1829      3
: 1120      1830      3
: 1121      1831      3
: 1122      1832      2
: 1123      1833      2
: 1124      1834      2
: 1125      1835      2
: 1126      1836      2
```


: 1127
: 11281837 2
1838 1 END;

```
OFFC 00000 TRIM_LEADING:
      5B      2C  90 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      : 1724
      5A      09  90 00005      MOV B      #32, BLANK                               : 1768
      50 00000000G 00 D0 00008      MOV B      #9, TAB                               : 1769
      51      60  90 0000F      MOV L      FDL$AB_STRING+4, R0                       : 1770
      50 00000000G 00 3C 00012      MOV B      (R0), TMP                               :
      59      50 D0 00019      MOVZWL     FDL$AB_STRING, R0                           : 1772
      5B      51  91 0001C      MOV L      R0, NEW_LEN                             :
      05      05 13 0001F      CMP B      TMP, BLANK                               : 1776
      5A      51  91 00021      BEQ L      1$
      53      53 12 00024      CMP B      TMP, TAB
      56      50 D0 00026 1$:      BNEQ      8$
      56      56 DD 00029      MOV L      R0, CUT_LEN                               : 1780
      00      01 FB 0002B      PUSHL     CUT_LEN                                   : 1781
      57      50 D0 00032      CALLS      #1, -FDL$$GET_VM
      50 00000000G 00 D0 00035      MOV L      R0, TMP_STR
      58      50 D0 0003C      MOV L      FDL$AB_STRING+4, R0                       : 1783
      60      56 28 0003F      MOV L      R0, STR
      59      56 D4 00043      MOV C3     CUT_LEN, (R0), (TMP_STR)                   : 1785
      50      59 D4 00045      CLRL      NEW_LEN                                   : 1787
      52      50 D4 00047      CLRL      J
      51      52  94 00047      CLRB      FLAG                                   : 1788
      51      50 A6 9E 00049      MOVAB     -1(R6), R1                               : 1789
      51      50 D1 0004D 2$:      J, R1                                           : 1791
      5B      1D 14 00050      BGTR      7$
      5B      6047 91 00052      CMP B      (J)[TMP_STR], BLANK                       : 1797
      06      06 13 00056      BEQ L      3$
      5A      6047 91 00058      CMP B      (J)[TMP_STR], TAB
      05      05 12 0005C      BNEQ      4$
      0A      52 E9 0005E 3$:      BLBC      FLAG, 6$
      03      03 11 00061      BRB
      52      01  90 00063 4$:      MOV B      #1, FLAG
      8948      6047 90 00066 5$:      MOV B      (J)[TMP_STR], (NEW_LEN)+[STR]
      50      50 D6 00068 6$:      INCL      J
      DE      DE 11 0006D      BRB
      7E      56 7D 0006F 7$:      BRB
      00      02 FB 00072      MOVQ      CUT_LEN, -(SP)
      50      59 D0 00079 8$:      CALLS      #2, -FDL$$FREE_VM
      04      04 0007C      MOV L      NEW_LEN, R0
      RET
      : 1836
      : 1838
```

; Routine Size: 125 bytes, Routine Base: _FDL\$CODE + 03E8

```
: 1130      1839 1 %SBTTL 'SET DATE TIME'
: 1131      1840 1 GLOBAL ROUTINE FDL$$SET_DATE_TIME =
: 1132      1841 1 ++
: 1133      1842 1
: 1134      1843 1 Functional Description:
: 1135      1844 1
: 1136      1845 1     Sets up the date/time quadword
: 1137      1846 1
: 1138      1847 1 Calling Sequence:
: 1139      1848 1
: 1140      1849 1     Called from the parse tables
: 1141      1850 1
: 1142      1851 1 Input Parameters:
: 1143      1852 1     none
: 1144      1853 1
: 1145      1854 1 Implicit Inputs:
: 1146      1855 1     none
: 1147      1856 1
: 1148      1857 1 Output Parameters:
: 1149      1858 1     none
: 1150      1859 1
: 1151      1860 1 Implicit Outputs:
: 1152      1861 1     none
: 1153      1862 1
: 1154      1863 1 Routine Value:
: 1155      1864 1     none
: 1156      1865 1
: 1157      1866 1 Side Effects:
: 1158      1867 1     none
: 1159      1868 1
: 1160      1869 1 --
: 1161      1870 1
: 1162      1871 2 BEGIN
: 1163      1872 2
: 1164      1873 2 TPARSE_ARGS;
: 1165      1874 2
: 1166      1875 2 LOCAL
: 1167      1876 2     TEMP_DESC      : DESC_BLK;
: 1168      1877 2
: 1169      1878 2 ! We must adjust the pointer so it points to the upcased buffer
: 1170      1879 2 !
: 1171      1880 2 TEMP_DESC [ DSC$W_LENGTH ] = .FDL$AB_STRING [ DSC$W_LENGTH ];
: 1172      1881 2 TEMP_DESC [ DSC$A_POINTER ] = .FDL$AB_STRING [ DSC$A_POINTER ] +
: 1173      1882 2     .FDL$GL_MAXLINE;
: 1174      1883 2
: 1175      1884 2 ! If there is an error signal it and return failure
: 1176      1885 2 !
: 1177      1886 2 IF NOT SYSSBINTIM( TEMP_DESC,FDL$AL_DATE_TIME )
: 1178      1887 2 THEN
: 1179      1888 2 BEGIN
: 1180      1889 2
: 1181      1890 2     BUILTIN CALLG;
: 1182      1891 2
: 1183      1892 2     TPARSE_BLOCK [ TPA$L_PARAM ] = FDL$_INV DATIM;
: 1184      1893 2
: 1185      1894 2     CALLG( .TPARSE_BLOCK,FDL$$$SYNTAX_ERROR );
: 1186      1895 2
```


FDLDRIVER
V04-000

VAX-11 FDL Utilities
SET_DATE_TIME

G 11
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 35
(14)

```
: 1187      1896  3      RETURN 0
: 1188      1897  3
: 1189      1898  2      END;
: 1190      1899  2
: 1191      1900  2      RETURN SS$_NORMAL
: 1192      1901  2
: 1193      1902  1      END;
```

```
                                0000 00000
                                08 C2 00002
                                00 B0 00005
04 AE 00000000G 00 00000000G 00 C1 0000C
                                00000000G 00 9F 00019
                                04 AE 9F 0001F
                                00000000G 00 02 FB 00022
                                11 50 E8 00029
                                20 AC 00000000G 8F D0 0002C
                                00000000V 00 6C FA 00034
                                50 04 11 0003B
                                01 D0 0003D 1$:
                                04 00040
                                50 D4 00041 2$:
                                04 00043
```

```
.ENTRY FDL$$SET_DATE_TIME, Save nothing      : 1840
SUBL2 #8, SP
MOVW FDL$AB_STRING, TEMP_DESC      : 1880
ADDL3 FDL$GL_MAXLINE, FDL$AB_STRING+4, -    : 1882
      TEMP_DESC+4
PUSHAB FDL$AL_DATE_TIME      : 1886
PUSHAB TEMP_DESC
CALLS #2, SY$$BINTIM
BLBS R0, 1$
MOVL #FDL$_INVDATIM, 32(TPARSE_BLOCK)      : 1892
CALLG (TPARSE_BLOCK), FDL$$SYNTAX_ERROR    : 1894
BRB 2$      : 1896
MOVL #1, R0      : 1900
RET
CLRL R0      : 1902
RET
```

; Routine Size: 68 bytes, Routine Base: _FDL\$CODE + 0465

```
: 1195      1903 1 %SBTTL 'SET COMMENT'
: 1196      1904 1 GLOBAL ROUTINE FDL$$SET_COMMENT =
: 1197      1905 1 ++
: 1198      1906 1
: 1199      1907 1 Functional Description:
: 1200      1908 1
: 1201      1909 1     Sets up the comment descriptor
: 1202      1910 1
: 1203      1911 1 Calling Sequence:
: 1204      1912 1
: 1205      1913 1     Called from the parse tables
: 1206      1914 1
: 1207      1915 1 Input Parameters:
: 1208      1916 1     none
: 1209      1917 1
: 1210      1918 1 Implicit Inputs:
: 1211      1919 1     none
: 1212      1920 1
: 1213      1921 1 Output Parameters:
: 1214      1922 1     none
: 1215      1923 1
: 1216      1924 1 Implicit Outputs:
: 1217      1925 1     none
: 1218      1926 1
: 1219      1927 1 Routine Value:
: 1220      1928 1     none
: 1221      1929 1
: 1222      1930 1 Side Effects:
: 1223      1931 1     none
: 1224      1932 1
: 1225      1933 1 --
: 1226      1934 1
: 1227      1935 2 BEGIN
: 1228      1936 2
: 1229      1937 2 TPARSE_ARGS;
: 1230      1938 2
: 1231      1939 2 ! The comment is the rest of the line
: 1232      1940 2 !
: 1233      1941 2 FDL$AB_COMMENT [ DSC$W_LENGTH ] = .TPARSE_BLOCK [ TPA$L_STRINGCNT ] + 1;
: 1234      1942 2 FDL$AB_COMMENT [ DSC$A_POINTER ] = .TPARSE_BLOCK [ TPA$[ _STRINGPTR ] ] - 1;
: 1235      1943 2
: 1236      1944 2 ! Adjust the pointer so that we are looking into the original input line
: 1237      1945 2 !
: 1238      1946 2 FDL$AB_COMMENT [ DSC$A_POINTER ] = .FDL$AB_COMMENT [ DSC$A_POINTER ] -
: 1239      1947 2     .FDL$GL_MAXLINE;
: 1240      1948 2
: 1241      1949 2 RETURN SSS_NORMAL
: 1242      1950 2
: 1243      1951 1 END;
```

```
FC  A2      08  AC      52 00000000G 0004 00000
                                00 9E 00002
                                01 A1 00009
```

```
.ENTRY FDL$$SET_COMMENT, Save R2
MOVAB  FDL$AB_COMMENT+4, R2
ADDW3  #1, 8(TPARSE_BLOCK), FDL$AB_COMMENT
```

```
: 1904
:
: 1941
```


FDLDRIVER
V04-000

VAX-11 FDL Utilities
SET_COMMENT

I 11
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 37
(15)

62 0C AC 01 C3 0000F
62 00000000G 00 C2 00014
50 01 D0 0001B
04 0001E

SUBL3 #1, 12(TPARSE_BLOCK), FDL\$AB_COMMENT+4
SUBL2 FDL\$GL_MAXLINE, FDL\$AB_COMMENT+4
MOVL #1, R0
RET

; 1942
; 1947
; 1949
; 1951

; Routine Size: 31 bytes, Routine Base: _FDL\$CODE + 04A9

```
: 1245      1952 1 %SBTTL 'SYNTAX_ERROR'
: 1246      1953 1 GLOBAL ROUTINE "FDL$$$SYNTAX_ERROR" =
: 1247      1954 1 ++
: 1248      1955 1
: 1249      1956 1 Functional Description:
: 1250      1957 1
: 1251      1958 1 Syntax error has two functions: If called with the argument fdl$_abkw
: 1252      1959 1 or fdl$_abprikw it checks if there has been an ambiguous keyword, if
: 1253      1960 1 there has been then it signals the error else it returns failure. If
: 1254      1961 1 it is called with some other error it is signaled and return is normal.
: 1255      1962 1
: 1256      1963 1 Calling Sequence:
: 1257      1964 1
: 1258      1965 1 Called from the parse tables
: 1259      1966 1
: 1260      1967 1 Can be called from a bliss routine by:
: 1261      1968 1
: 1262      1969 1 BUILTIN CALLG;
: 1263      1970 1
: 1264      1971 1 CALLG( tparse_block,FDL$$$SYNTAX_ERROR )
: 1265      1972 1
: 1266      1973 1 Input Parameters:
: 1267      1974 1
: 1268      1975 1 Error code in the tpa$l_param field of the tparse_block
: 1269      1976 1
: 1270      1977 1 Implicit Inputs:
: 1271      1978 1 none
: 1272      1979 1
: 1273      1980 1 Output Parameters:
: 1274      1981 1 none
: 1275      1982 1
: 1276      1983 1 Implicit Outputs:
: 1277      1984 1 none
: 1278      1985 1
: 1279      1986 1 Routine Value:
: 1280      1987 1
: 1281      1988 1 ss$_normal or 0 (see above)
: 1282      1989 1
: 1283      1990 1 Side Effects:
: 1284      1991 1
: 1285      1992 1 Signals an error
: 1286      1993 1
: 1287      1994 1 --
: 1288      1995 1
: 1289      1996 2 BEGIN
: 1290      1997 2
: 1291      1998 2 TPARSE_ARGS;
: 1292      1999 2
: 1293      2000 2 LOCAL STATUS : LONG;
: 1294      2001 2
: 1295      2002 2 BIND CODE = STATUS : BLOCK [ 4,BYTE ];
: 1296      2003 2
: 1297      2004 2 ! Get the error code passed to us by the parse tables
: 1298      2005 2 !
: 1299      2006 2 STATUS = .TPARSE_BLOCK [ TPA$L_PARAM ];
: 1300      2007 2
: 1301      2008 2 ! If this is a ambiguity check and there is none return failure
```


			0000	00000	.ENTRY	FDL\$\$\$SYNTAX ERROR, Save nothing
	50	20	AC	D0 00002	MOVL	32(TPARSE BLOCK), STATUS
00000000G	8F		50	D1 00006	CML	STATUS, #FDL\$_ABKW
			09	13 0000D	BEQL	1\$
00000000G	8F		50	D1 0000F	CML	STATUS, #FDL\$_ABPRIKW
			04	12 00016	BNEQ	2\$
	2C	06	AC	E9 00018	BLBC	6(TPARSE BLOCK), 4\$
	03		00	ED 0001C	CMPZV	#0, #3, CODE, #3
			07	13 00021	BEQL	3\$
00000000G	00		08	88 00023	BISB2	#8, FDL\$AB CTRL
	7E	08	AC	7D 0002A	MOVQ	8(TPARSE BLOCK), -(SP)
	7E	10	AC	7D 0002E	MOVQ	16(TPARSE BLOCK), -(SP)
		00000000G	00	DD 00032	PUSHL	FDL\$GL_STAMTNUM
			05	DD 00038	PUSHL	#5
		20	AC	DD 0003A	PUSHL	32(TPARSE BLOCK)
00000000G	00		07	FB 0003D	CALLS	#7, LIB\$\$SIGNAL
	50		01	D0 00044	MOVL	#1, R0
			04	00047	RET	

1953
2006
2011

2012
2018

2023
2037
2035
2034
2033

2040

FDLDRIVER
V04-000

VAX-11 FDL Utilities
SYNTAX_ERROR

L 11
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 40
(16)

50 D4 00048 4\$: CLRL R0
04 0004A RET

; 2042
;

; Routine Size: 75 bytes, Routine Base: _FDL\$CODE + 04C8


```
: 1337      2043 1 %SBTTL 'NEGATE'
: 1338      2044 1 GLOBAL ROUTINE FDL$$NEGATE : NOVALUE =
: 1339      2045 1 ++
: 1340      2046 1
: 1341      2047 1 Functional Description:
: 1342      2048 1
: 1343      2049 1     Produces the negative version of a number
: 1344      2050 1
: 1345      2051 1 Calling Sequence:
: 1346      2052 1
: 1347      2053 1     Called from the parse tables
: 1348      2054 1
: 1349      2055 1 Input Parameters:
: 1350      2056 1     none
: 1351      2057 1
: 1352      2058 1 Implicit Inputs:
: 1353      2059 1     none
: 1354      2060 1
: 1355      2061 1 Output Parameters:
: 1356      2062 1     none
: 1357      2063 1
: 1358      2064 1 Implicit Outputs:
: 1359      2065 1     none
: 1360      2066 1
: 1361      2067 1 Routine Value:
: 1362      2068 1
: 1363      2069 1     none
: 1364      2070 1
: 1365      2071 1 Side Effects:
: 1366      2072 1     none
: 1367      2073 1
: 1368      2074 1 --
: 1369      2075 1
: 1370      2076 2 BEGIN
: 1371      2077 2
: 1372      2078 2 TPARSE_ARGS;
: 1373      2079 2
: 1374      2080 2 | Just negate the number
: 1375      2081 2 |
: 1376      2082 2 FDL$GL_NUMBER = -.FDL$GL_NUMBER;
: 1377      2083 2
: 1378      2084 2 RETURN
: 1379      2085 2
: 1380      2086 1 END;
```

```
                    0004 00000
52 00000000G 00 9E 00002
62          62 CE 00009
                    04 0000C
```

```
.ENTRY FDL$$NEGATE, Save R2
MOVAB FDL$GL_NUMBER, R2
MNEGL FDL$GL_NUMBER, FDL$GL_NUMBER
RET
```

```
: 2044
:
: 2082
: 2086
```

; Routine Size: 13 bytes, Routine Base: _FDL\$CODE + 0513

```
: 1382      2087 1 %SBTTL 'SET BLANK'
: 1383      2088 1 GLOBAL ROUTINE FDL$$SET_BLANK : NOVALUE =
: 1384      2089 1 ++
: 1385      2090 1
: 1386      2091 1 Functional Description:
: 1387      2092 1
: 1388      2093 1     Sets the Tparse blanks flag to allow parsing of blanks
: 1389      2094 1
: 1390      2095 1 Calling Sequence:
: 1391      2096 1
: 1392      2097 1     Called from the parse tables
: 1393      2098 1
: 1394      2099 1 Input Parameters:
: 1395      2100 1     none
: 1396      2101 1
: 1397      2102 1 Implicit Inputs:
: 1398      2103 1     none
: 1399      2104 1
: 1400      2105 1 Output Parameters:
: 1401      2106 1     none
: 1402      2107 1
: 1403      2108 1 Implicit Outputs:
: 1404      2109 1     none
: 1405      2110 1
: 1406      2111 1 Routine Value:
: 1407      2112 1
: 1408      2113 1     none
: 1409      2114 1
: 1410      2115 1 Side Effects:
: 1411      2116 1     none
: 1412      2117 1
: 1413      2118 1 --
: 1414      2119 1
: 1415      2120 2 BEGIN
: 1416      2121 2
: 1417      2122 2 TPARSE_ARGS;
: 1418      2123 2
: 1419      2124 2 ! Just set the flag
: 1420      2125 2 !
: 1421      2126 2 TPARSE_BLOCK [ TPA$V_BLANKS ] = _SET;
: 1422      2127 2
: 1423      2128 2 RETURN
: 1424      2129 2
: 1425      2130 1 END;
```

```
04 AC          0000 00000
01 88 00002
04 00006
```

```
.ENTRY FDL$$SET_BLANK, Save nothing
BISB2 #1, 4(TPARSE_BLOCK)
RET
```

```
: 2088
: 2126
: 2130
```

; Routine Size: 7 bytes, Routine Base: _FDL\$CODE + 0520


```
: 1427      2131 1 %SBTTL 'CLR BLANK'
: 1428      2132 1 GLOBAL ROUTINE FDL$$CLR_BLANK : NOVALUE =
: 1429      2133 1 ++
: 1430      2134 1
: 1431      2135 1 Functional Description:
: 1432      2136 1
: 1433      2137 1     Clears the Tparse blanks flag
: 1434      2138 1
: 1435      2139 1 Calling Sequence:
: 1436      2140 1
: 1437      2141 1     Called from the parse tables
: 1438      2142 1
: 1439      2143 1 Input Parameters:
: 1440      2144 1     none
: 1441      2145 1
: 1442      2146 1 Implicit Inputs:
: 1443      2147 1     none
: 1444      2148 1
: 1445      2149 1 Output Parameters:
: 1446      2150 1     none
: 1447      2151 1
: 1448      2152 1 Implicit Outputs:
: 1449      2153 1     none
: 1450      2154 1
: 1451      2155 1 Routine Value:
: 1452      2156 1
: 1453      2157 1     none
: 1454      2158 1
: 1455      2159 1 Side Effects:
: 1456      2160 1     none
: 1457      2161 1
: 1458      2162 1 --
: 1459      2163 1
: 1460      2164 2 BEGIN
: 1461      2165 2
: 1462      2166 2 TPARSE_ARGS;
: 1463      2167 2
: 1464      2168 2 ! Just clear the flag
: 1465      2169 2 !
: 1466      2170 2 TPARSE_BLOCK [ TPA$V_BLANKS ] = _CLEAR;
: 1467      2171 2
: 1468      2172 2 RETURN
: 1469      2173 2
: 1470      2174 1 END;
```

```
04 AC          0000 00000
01 8A 00002
04 00006
```

```
.ENTRY FDL$$CLR_BLANK, Save nothing
BICB2 #1, 4(TPARSE_BLOCK)
RET
```

```
: 2132
: 2170
: 2174
```

; Routine Size: 7 bytes, Routine Base: _FDL\$CODE + 0527

```
: 1472 2175 1 %SBTTL 'ERROR_CHK'
: 1473 2176 1 GLOBAL ROUTINE -FDL$$ERROR_CHK =
: 1474 2177 1 ++
: 1475 2178 1
: 1476 2179 1 Functional Description:
: 1477 2180 1
: 1478 2181 1 Does a check if there was a warning
: 1479 2182 1
: 1480 2183 1 Calling Sequence:
: 1481 2184 1
: 1482 2185 1 Called from the parse tables
: 1483 2186 1
: 1484 2187 1 Input Parameters:
: 1485 2188 1 none
: 1486 2189 1
: 1487 2190 1 Implicit Inputs:
: 1488 2191 1 none
: 1489 2192 1
: 1490 2193 1 Output Parameters:
: 1491 2194 1 none
: 1492 2195 1
: 1493 2196 1 Implicit Outputs:
: 1494 2197 1 none
: 1495 2198 1
: 1496 2199 1 Routine Value:
: 1497 2200 1
: 1498 2201 1 Value of fdl$ab_ctrl [ fdl$v_warning ]
: 1499 2202 1
: 1500 2203 1 Side Effects:
: 1501 2204 1 none
: 1502 2205 1
: 1503 2206 1 --
: 1504 2207 1
: 1505 2208 2 BEGIN
: 1506 2209 2
: 1507 2210 2 TPARSE_ARGS;
: 1508 2211 2
: 1509 2212 2 ! If there is a warning return true else fail
: 1510 2213 2 !
: 1511 2214 2 RETURN .FDL$AB_CTRL [ FDL$V_WARNING ]
: 1512 2215 2
: 1513 2216 1 END;
```

```
50 00000000G 00 01 0000 00000
03 EF 00002
04 0000B
```

```
.ENTRY FDL$$ERROR_CHK, Save nothing
EXTZV #3, #1, FDL$AB_CTRL, R0
RET
```

```
: 2176
: 2214
: 2216
```

; Routine Size: 12 bytes, Routine Base: _FDL\$CODE + 052E

; 1514 2217 1


```
: 1516      2218 1 %SBTTL 'FDL$$READ_ERROR'
: 1517      2219 1 GLOBAL ROUTINE FDL$$READ_ERROR : NOVALUE =
: 1518      2220 1 ++
: 1519      2221 1
: 1520      2222 1 Functional Description:
: 1521      2223 1
: 1522      2224 1 This routine will signal an rms error and stop execution if the RMS
: 1523      2225 1 error is NOT end of file. It is to be used for detecting errors
: 1524      2226 1 during rms $GETs or $READs.
: 1525      2227 1
: 1526      2228 1 Calling Sequence:
: 1527      2229 1
: 1528      2230 1 This routine is call as an AST by RMS
: 1529      2231 1
: 1530      2232 1 Input Parameters:
: 1531      2233 1
: 1532      2234 1 AST argument block which has a pointer to a RAB
: 1533      2235 1
: 1534      2236 1 Implicit Inputs:
: 1535      2237 1 none
: 1536      2238 1
: 1537      2239 1 Output Parameters:
: 1538      2240 1 none
: 1539      2241 1
: 1540      2242 1 Implicit Outputs:
: 1541      2243 1 none
: 1542      2244 1
: 1543      2245 1 Routine Value:
: 1544      2246 1 none
: 1545      2247 1
: 1546      2248 1 Routines Called:
: 1547      2249 1
: 1548      2250 1 SIGNAL_STOP
: 1549      2251 1
: 1550      2252 1 Side Effects:
: 1551      2253 1 none
: 1552      2254 1
: 1553      2255 1 --
: 1554      2256 1
: 1555      2257 2 BEGIN
: 1556      2258 2
: 1557      2259 2 BUILTIN
: 1558      2260 2 AP;
: 1559      2261 2
: 1560      2262 2 BIND
: 1561      2263 2 AST_BLOCK = AP : REF VECTOR [ ,LONG ];
: 1562      2264 2
: 1563      2265 2 LOCAL
: 1564      2266 2 RAB : REF BLOCK [ ,BYTE ],
: 1565      2267 2 FAB : REF BLOCK [ ,BYTE ],
: 1566      2268 2 NAM : REF BLOCK [ ,BYTE ];
: 1567      2269 2
: 1568      2270 2 ! Get the rab (Pointer to by the second ast parameter)
: 1569      2271 2 !
: 1570      2272 2 RAB = .AST_BLOCK [ 1 ];
: 1571      2273 2
: 1572      2274 2 ! If this is only an end of file then return
```

```
: 1573      2275 2      !
: 1574      2276 2      !IF .RAB [ RAB$L_STS ] EQLU RMSS_EOF
: 1575      2277 2      THEN
: 1576      2278 2      RETURN;
: 1577      2279 2
: 1578      2280 2      ! Now get the fab it points to
: 1579      2281 2
: 1580      2282 2      FAB = .RAB [ RAB$L_FAB ];
: 1581      2283 2
: 1582      2284 2      ! Get the name block
: 1583      2285 2
: 1584      2286 2      NAM = .FAB [ FAB$L_NAM ];
: 1585      2287 2
: 1586      2288 2      ! Signal the FDL error with the best file name string
: 1587      2289 2
: 1588      2290 2      ! First try the resultant string
: 1589      2291 2
: 1590      2292 2      IF .NAM [ NAM$B_RSL ] NEQU 0
: 1591      2293 2      THEN
: 1592      2294 2          BEGIN
: 1593      2295 2              STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
: 1594      2296 2              STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$B_RSA ];
: 1595      2297 2          END
: 1596      2298 2
: 1597      2299 2      ! Next try the expanded string
: 1598      2300 2
: 1599      2301 2      ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
: 1600      2302 2      THEN
: 1601      2303 2          BEGIN
: 1602      2304 2              STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
: 1603      2305 2              STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$B_ESA ];
: 1604      2306 2          END
: 1605      2307 2
: 1606      2308 2      ! If all else fails use the name string
: 1607      2309 2
: 1608      2310 2      ELSE
: 1609      2311 2          BEGIN
: 1610      2312 2              STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
: 1611      2313 2              STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$B_FNA ];
: 1612      2314 2          END;
: 1613      2315 2
: 1614      2316 2      SIGNAL_STOP( .RAB [ RAB$L_CTX ],1,STRING_DESC,
: 1615      2317 2          .RAB [ FAB$L_STS ],.RAB [ FAB$L_STV ] )
: 1616      2318 2
: 1617      2319 1      END;
```

```
0001827A 53 00000000' 00 000C 00000
          52      04 AC 9E 00002
          8F      08 A2 D0 00009
          51      3C A2 D1 0000D
          50      28 A3 13 00015
          51      3C A2 D0 00017
          50      28 A1 D0 0001B
```

```
.ENTRY FDL$$READ_ERROR, Save R2,R3
MOVAB STRING_DESC, R3
MOVL 4(AST_BLOCK), RAB
CML 8(RAB), #98938
BEQL 4$
MOVL 60(RAB), FAB
MOVL 40(FAB), NAM
```

```
: 2219
: 2272
: 2276
: 2282
: 2286
```


FDLDRIVER
V04-000

VAX-11 FDL Utilities
FDL\$\$READ_ERROR

F 12
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 47
(21)

		03	A0	95	0001F	TSTB	3(NAM)	: 2292
			0B	13	00022	BEQL	1\$: 2295
04	63	03	A0	9B	00024	MOVZBW	3(NAM), STRING_DESC	: 2296
	A3	04	A0	D0	00028	MOVL	4(NAM), STRING_DESC+4	: 2301
			19	11	0002D	BRB	3\$: 2304
		0B	A0	95	0002F	TSTB	11(NAM)	: 2305
			0B	13	00032	BEQL	2\$: 2312
04	63	0B	A0	9B	00034	MOVZBW	11(NAM), STRING_DESC	: 2313
	A3	0C	A0	D0	00038	MOVL	12(NAM), STRING_DESC+4	: 2317
			09	11	0003D	BRB	3\$: 2316
04	63	34	A1	9B	0003F	MOVZBW	52(FAB), STRING_DESC	: 2319
	A3	2C	A1	D0	00043	MOVL	44(FAB), STRING_DESC+4	: 2319
	7E	08	A2	7D	00048	MOVQ	8(RAB), -(SP)	: 2319
			53	DD	0004C	PUSHL	R3	: 2319
			01	DD	0004E	PUSHL	#1	: 2319
		18	A2	DD	00050	PUSHL	24(RAB)	: 2319
000000J0G	00		05	FB	00053	CALLS	#5, LIB\$STOP	: 2319
			04	0005A	4\$:	RET		: 2319

; Routine Size: 91 bytes, Routine Base: _FDL\$CODE + 053A

; 1618 2320 1

```
: 1620      2321 1 %SBTTL 'FDL$$RMS_ERROR'
: 1621      2322 1 GLOBAL ROUTINE FDL$$RMS_ERROR : NOVALUE =
: 1622      2323 1 ++
: 1623      2324 1
: 1624      2325 1 Functional Description:
: 1625      2326 1
: 1626      2327 1 This routine will signal and rms error and stop execution. It is
: 1627      2328 1 to be primarily used for detecting errors during asynchronous operations
: 1628      2329 1
: 1629      2330 1 Calling Sequence:
: 1630      2331 1
: 1631      2332 1 This routine is call as an AST by RMS
: 1632      2333 1
: 1633      2334 1 Input Parameters:
: 1634      2335 1
: 1635      2336 1 AST argument block which has a pointer to a rms block
: 1636      2337 1
: 1637      2338 1 Implicit Inputs:
: 1638      2339 1 none
: 1639      2340 1
: 1640      2341 1 Output Parameters:
: 1641      2342 1 none
: 1642      2343 1
: 1643      2344 1 Implicit Outputs:
: 1644      2345 1 none
: 1645      2346 1
: 1646      2347 1 Routine Value:
: 1647      2348 1 none
: 1648      2349 1
: 1649      2350 1 Routines Called:
: 1650      2351 1
: 1651      2352 1 SIGNAL_STOP
: 1652      2353 1
: 1653      2354 1 Side Effects:
: 1654      2355 1 none
: 1655      2356 1
: 1656      2357 1 --
: 1657      2358 1
: 1658      2359 2 BEGIN
: 1659      2360 2
: 1660      2361 2 BUILTIN AP;
: 1661      2362 2
: 1662      2363 2 BIND
: 1663      2364 2 AST_BLOCK = AP : REF VECTOR [ ,LONG ];
: 1664      2365 2
: 1665      2366 2 LOCAL
: 1666      2367 2 RMS_BLOCK : REF BLOCK [ ,BYTE ];
: 1667      2368 2
: 1668      2369 2 ! Get the rms control block (second argument in the block)
: 1669      2370 2
: 1670      2371 2 RMS_BLOCK = .AST_BLOCK [ 1 ];
: 1671      2372 2
: 1672      2373 2 ! NOTE: We use the RAB$x_zzz codes but they are valid for the FAB as well
: 1673      2374 2
: 1674      2375 2 ! Signal the FDL error
: 1675      2376 2
: 1676      2377 2 SIGNAL_STOP( .RMS_BLOCK [ RAB$L_CTX ],
```


FDLDRIVER
V04-000

VAX-11 FDL Utilities
FDL\$\$RMS_ERROR

H 12
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 49
(22)

: 1677
: 1678
: 1679

2378 2
2379 2
2380 1

END;

.RMS_BLOCK [RAB\$L_STS],.RMS_BLOCK [RAB\$L_STV])

50 04 0000 00000
7E 08 AC DO 00002
18 A0 7D 00006
A0 DD 0000A
03 FB 0000D
04 00014

00000000G 00

.ENTRY FDL\$\$RMS_ERROR, Save nothing
MOVL 4(AST_BLOCK), RMS_BLOCK
MOVQ 8(RMS_BLOCK), -(SP)
PUSHL 24(RMS_BLOCK)
CALLS #3, LIB\$STOP
RET

: 2322
: 2371
: 2378
: 2377
: 2380

; Routine Size: 21 bytes, Routine Base: _FDL\$CODE + 0595

; 1680 2381 1

```
: 1682 2382 1 %SBTTL 'FDL$$RMS_OPEN_ERROR'
: 1683 2383 1 GLOBAL ROUTINE FDL$$RMS_OPEN_ERROR : NOVALUE =
: 1684 2384 1 ++
: 1685 2385 1
: 1686 2386 1 Functional Description:
: 1687 2387 1
: 1688 2388 1 This routine will signal an rms error and stop execution. It is
: 1689 2389 1 to be primarily used for detecting errors during file opens.
: 1690 2390 1
: 1691 2391 1 Calling Sequence:
: 1692 2392 1
: 1693 2393 1 This routine is call as an AST by RMS
: 1694 2394 1
: 1695 2395 1 Input Parameters:
: 1696 2396 1
: 1697 2397 1 AST argument block which has a pointer to a FAB
: 1698 2398 1
: 1699 2399 1 Implicit Inputs:
: 1700 2400 1 none
: 1701 2401 1
: 1702 2402 1 Output Parameters:
: 1703 2403 1 none
: 1704 2404 1
: 1705 2405 1 Implicit Outputs:
: 1706 2406 1 none
: 1707 2407 1
: 1708 2408 1 Routine Value:
: 1709 2409 1 none
: 1710 2410 1
: 1711 2411 1 Routines Called:
: 1712 2412 1
: 1713 2413 1 SIGNAL_STOP
: 1714 2414 1
: 1715 2415 1 Side Effects:
: 1716 2416 1 none
: 1717 2417 1
: 1718 2418 1 --
: 1719 2419 1
: 1720 2420 2 BEGIN
: 1721 2421 2
: 1722 2422 2 BUILTIN
: 1723 2423 2 AP;
: 1724 2424 2
: 1725 2425 2 BIND
: 1726 2426 2 AST_BLOCK = AP : REF VECTOR [ ,LONG ];
: 1727 2427 2
: 1728 2428 2 LOCAL
: 1729 2429 2 FAB : REF BLOCK [ ,BYTE ],
: 1730 2430 2 NAM : REF BLOCK [ ,BYTE ];
: 1731 2431 2
: 1732 2432 2 ! Get the fab (Pointer to by the second ast parameter)
: 1733 2433 2 !
: 1734 2434 2 FAB = .AST_BLOCK [ 1 ];
: 1735 2435 2
: 1736 2436 2 ! If this is really a RAB (from a connect) then get the fab it points to
: 1737 2437 2 !
: 1738 2438 2 IF .FAB [ FAB$B_BID ] EQLU RAB$C_BID
```



```

: 1739      2439 2      THEN
: 1740      2440 2          FAB = .FAB [ RAB$L_FAB ];      ! This looks strange but it's ok!
: 1741      2441 2
: 1742      2442 2      ! Get the name block
: 1743      2443 2      !
: 1744      2444 2      NAM = .FAB [ FAB$L_NAM ];
: 1745      2445 2
: 1746      2446 2      ! Signal the FDL error with the best file name string
: 1747      2447 2      !
: 1748      2448 2      ! First try the resultant string
: 1749      2449 2      !
: 1750      2450 2      IF .NAM [ NAM$B_RSL ] NEQU 0
: 1751      2451 2      THEN
: 1752      2452 2          BEGIN
: 1753      2453 2              STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
: 1754      2454 2              STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[ _RSA ] ]
: 1755      2455 2          END
: 1756      2456 2
: 1757      2457 2      ! Next try the expanded string
: 1758      2458 2      !
: 1759      2459 2      ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
: 1760      2460 2      THEN
: 1761      2461 2          BEGIN
: 1762      2462 2              STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
: 1763      2463 2              STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[ _ESA ] ]
: 1764      2464 2          END
: 1765      2465 2
: 1766      2466 2      ! If all else fails use the name string
: 1767      2467 2      !
: 1768      2468 2      ELSE
: 1769      2469 2          BEGIN
: 1770      2470 2              STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
: 1771      2471 2              STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$[ _FNA ] ]
: 1772      2472 2          END;
: 1773      2473 2
: 1774      2474 2      SIGNAL_STOP( .FAB [ RAB$L_CTX ],1,STRING_DESC,
: 1775      2475 2          .FAB [ FAB$L_STS ],.FAB [ FAB$L_STV ] )
: 1776      2476 2
: 1777      2477 1      END;

```

			0004 00000	.ENTRY	FDL\$SRMS OPEN ERROR, Save R2	: 2383
52	00000000'	00	9E 00002	MOVAB	STRING_DESC, R2	: 2434
51	04	AC	D0 00009	MOVL	4(AST_BLOCK), FAB	: 2438
01		61	91 0000D	CMPB	(FAB), #1	
		04	12 00010	BNEQ	1\$	
51	3C	A1	D0 00012	MOVL	60(FAB), FAB	: 2440
50	28	A1	D0 00016 1\$:	MOVL	40(FAB), NAM	: 2444
	03	A0	95 0001A	TSTB	3(NAM)	: 2450
		0B	13 0001D	BEQL	2\$	
62	03	A0	9B 0001F	MOVZBW	3(NAM), STRING_DESC	: 2453
04	A2	04	A0 D0 00023	MOVL	4(NAM), STRING_DESC+4	: 2454
		19	11 00028	BRB	4\$	
		0B	A0 95 0002A 2\$:	TSTB	11(NAM)	: 2459

FDLDRIVER
V04-000

VAX-11 FDL Utilities
FDL\$SRMS_OPEN_ERROR

K 12
16-Sep-1984 01:47:45
14-Sep-1984 12:31:17

VAX-11 Bliss-32 V4.0-742
[FDL.SRC]FDLDRIVER.B32;1

Page 52
(23)

			0B	13	0002D	BEQL	3\$		
			A0	9B	0002F	MOVZBW	11(NAM),	STRING_DESC	2462
04	62	0B	A0	D0	00033	MOVL	12(NAM),	STRING_DESC+4	2463
	A2	0C	09	11	00038	BRB	4\$		
			A1	9B	0003A	MOVZBW	52(FAB),	STRING_DESC	2470
04	62	34	A1	D0	0003E	MOVL	44(FAB),	STRING_DESC+4	2471
	A2	2C	A1	7D	00043	MOVQ	8(FAB),	-(SP)	2475
	7E	08	52	DD	00047	PUSHL	R2		2474
			01	DD	00049	PUSHL	#1		
			A1	DD	0004B	PUSHL	24(FAB)		
00000000G	00	18	05	FB	0004E	CALLS	#5, LIB\$STOP		
			04	00055	RET				2477

; Routine Size: 86 bytes, Routine Base: _FDL\$CODE + 05AA

: 1778 2478 1
: 1779 2479 0 END ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
_FDL\$OWN	12	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_FDL\$CODE	1536	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	39	0	581	00:01.0

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/LIS=LIS\$:FDLDRIVER/OBJ=OBJ\$:FDLDRIVER MSRC\$:FDLDRIVER/UPDATE=(ENHS\$:FDLDRIVER)

: Size: 1536 code + 12 data bytes
: Run Time: 00:36.1
: Elapsed Time: 02:08.7
: Lines/CPU Min: 4125
: Lexemes/CPU-Min: 21518
: Memory Used: 175 pages

FDLDRIVER
V04-000

VAX-11 FDL Utilities
FDL\$SRMS_OPEN_ERROR

L 12
16-Sep-1984 01:47:45

VAX-11 Bliss-32 V4.0-742

Page 53

; Compilation Complete

0176 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY